AIM:- Study different python libraries.

1. Pandas Library:

- Load a dataset (Iris dataset: https://www.kagale.com/datasets/uciml/iris) using pandas and display the first few rows to understand its structure.

```
import pandas as pd
df = pd.read_csv("D:\\5th Sem\\LAB\\ML\\Day-2\\Iris.csv")
print("First 5 rows of the Iris dataset:-\n", df.head())
```

Output:

```
First 5 rows of the Iris dataset:-
   Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm      Species
0   1            5.1           3.5            1.4           0.2  Iris-setosa
1   2            4.9           3.0            1.4           0.2  Iris-setosa
2   3            4.7           3.2            1.3           0.2  Iris-setosa
3   4            4.6           3.1            1.5           0.2  Iris-setosa
4   5            5.0           3.6            1.4           0.2  Iris-setosa
```

- Calculate basic statistics (mean, median, standard deviation, etc.) for a numerical column in the dataset.

```
result = df.drop(['Id', 'Species'], axis=1)
print("After removing the unnecessary columns:-\n", result)
print("Basic Statistics of the dataset:-\n", result.describe())
```

Output:

```
After removing the unnecessary columns:-
      SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
0               5.1           3.5            1.4           0.2
1               4.9           3.0            1.4           0.2
2               4.7           3.2            1.3           0.2
3               4.6           3.1            1.5           0.2
4               5.0           3.6            1.4           0.2
..              ...           ...            ...           ...
145             6.7           3.0            5.2           2.3
146             6.3           2.5            5.0           1.9
147             6.5           3.0            5.2           2.0
148             6.2           3.4            5.4           2.3
149             5.9           3.0            5.1           1.8

[150 rows x 4 columns]
```

```
Basic Statistics of the dataset:-
       SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
count     150.000000    150.000000     150.000000    150.000000
mean        5.843333      3.054000       3.758667      1.198667
std         0.828066      0.433594       1.764420      0.763161
min         4.300000      2.000000       1.000000      0.100000
25%         5.100000      2.800000       1.600000      0.300000
50%         5.800000      3.000000       4.350000      1.300000
75%         6.400000      3.300000       5.100000      1.800000
max         7.900000      4.400000       6.900000      2.500000
```

- Perform data filtering to extract rows based on specific conditions (e.g. SepalLengthCm>5.0)

```
condition = df['SepalLengthCm'] > 5.0
filtered_iris = df[condition]
print("The rows having SepalLengthCm > 5.0 is:-\n", filtered_iris)
```

Output:

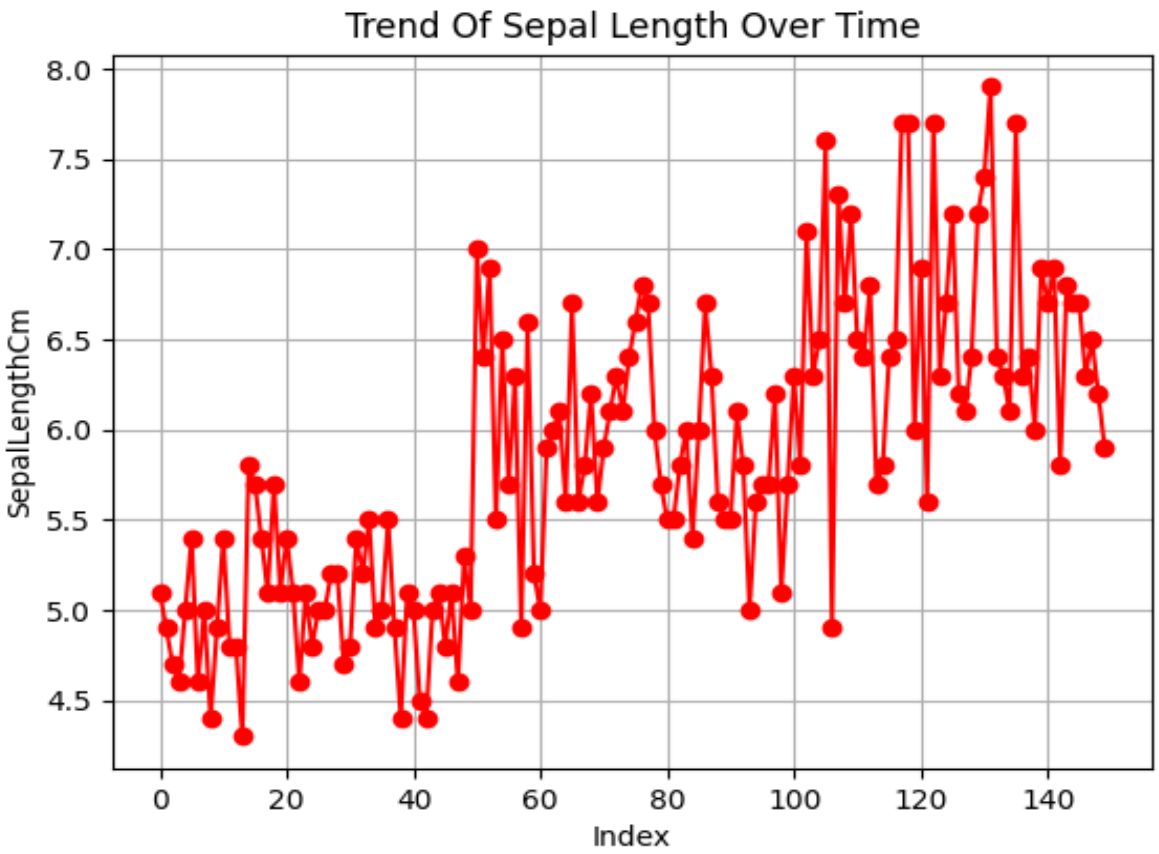
```
The rows having SepalLengthCm > 5.0 is:-
       Id  SepalLengthCm  ...  PetalWidthCm         Species
0       1            5.1  ...           0.2     Iris-setosa
5       6            5.4  ...           0.4     Iris-setosa
10     11            5.4  ...           0.2     Iris-setosa
14     15            5.8  ...           0.2     Iris-setosa
15     16            5.7  ...           0.4     Iris-setosa
..    ...            ...  ...           ...             ...
145   146            6.7  ...           2.3  Iris-virginica
146   147            6.3  ...           1.9  Iris-virginica
147   148            6.5  ...           2.0  Iris-virginica
148   149            6.2  ...           2.3  Iris-virginica
149   150            5.9  ...           1.8  Iris-virginica

[118 rows x 6 columns]
```

2. **Matplotlib Library:**

- Create a line plot to visualize the trend of a numerical variable over time

```
# Import matplotlib.pyplot as plt and df from Pandas.py
import matplotlib.pyplot as plt
from Pandas import df
plt.plot(df['SepalLengthCm'], marker='o', linestyle='-', color='r')
plt.xlabel('Index')
plt.ylabel('SepalLengthCm')
plt.title(Trend Of Sepal Length Over Time')
plt.grid(True)
plt.show()
```

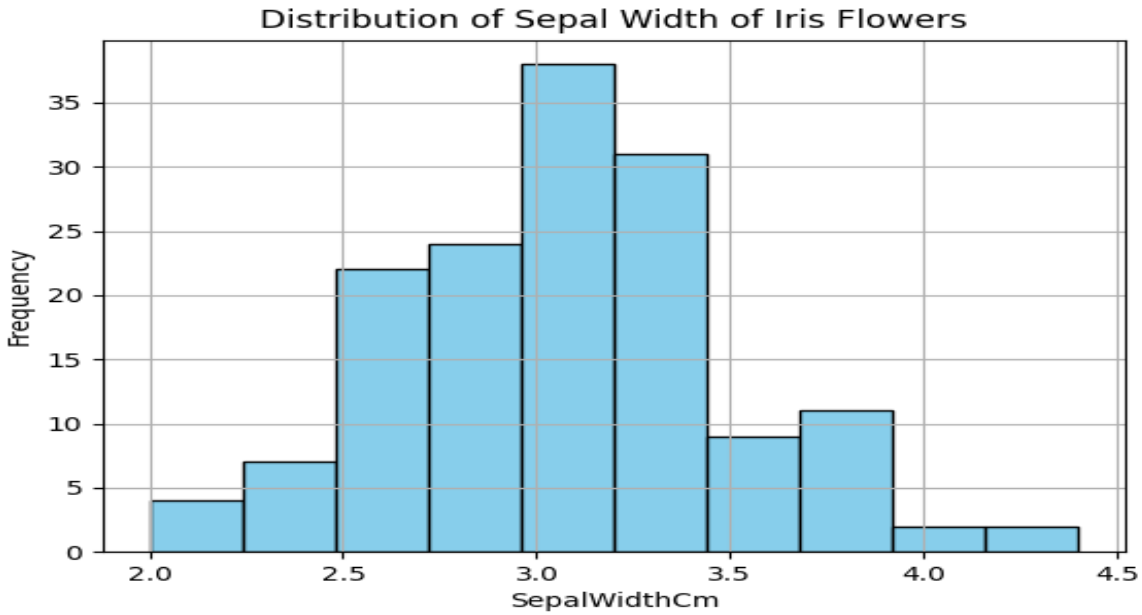
Output:



- Generate a histogram to understand the distribution of a numerical variable in the dataset.

```
plt.hist(df['SepalWidthCm'], edgecolor='black', color='skyblue')
plt.xlabel('SepalWidthCm')
plt.ylabel('Frequency')
plt.title('Distribution of Sepal Width of Iris Flowers')
plt.grid(True)
plt.show()
```

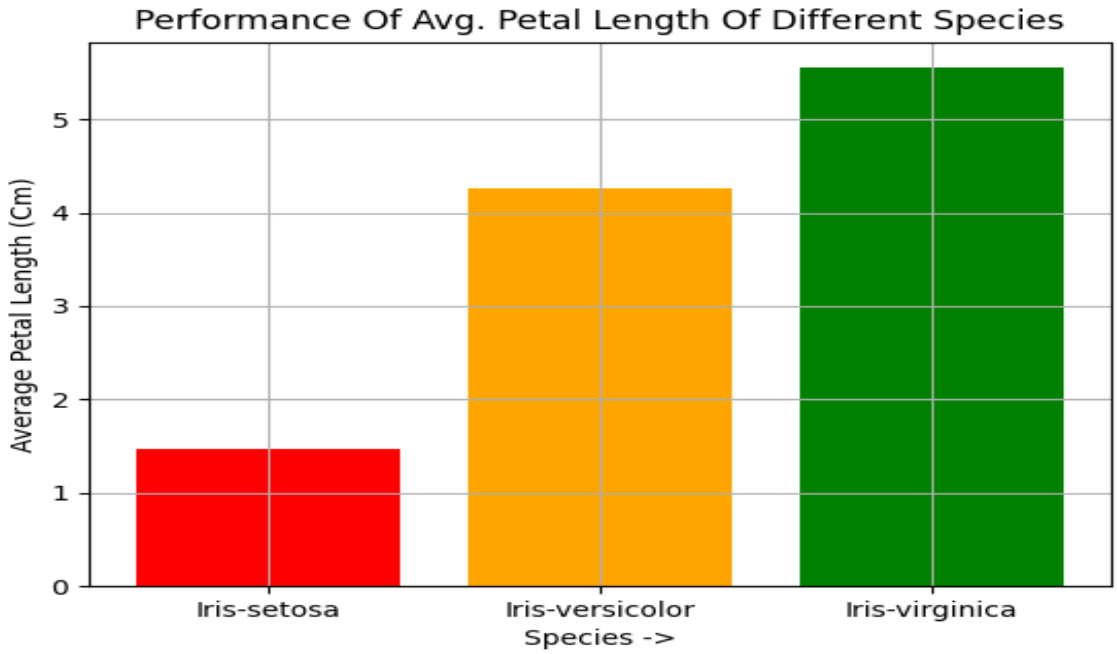
Output:



- Create a bar chart to compare the performance of different categories.

```
avg_petal_length = df.groupby('Species')['PetalLengthCm'].mean()
species_names = avg_petal_length.index
plt.bar(species_names, avg_petal_length, color=['r', 'orange', 'g'])
plt.xlabel('Species ->')
plt.ylabel('Average Petal Length (Cm)')
plt.title('Performance Of Avg. Petal Length Of Different Species')
plt.grid(True)
plt.show()
```

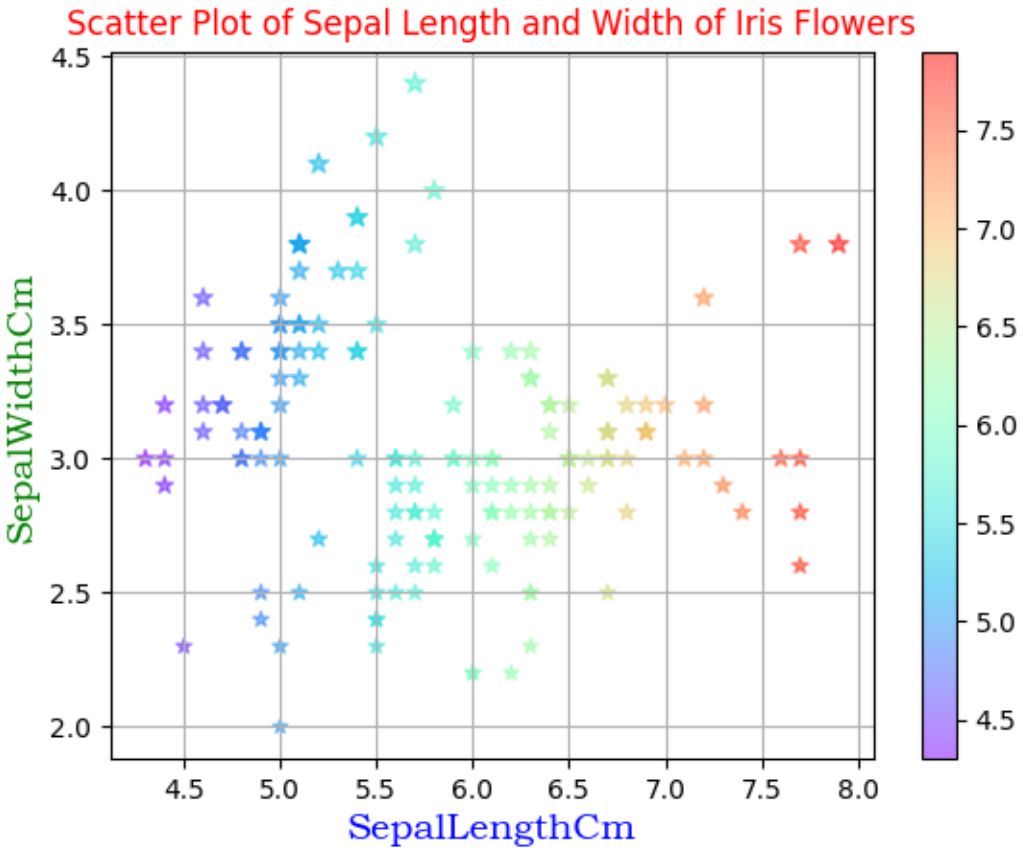
Output:



- Plot a scatter plot to explore the relationship between two numerical variables.
- Customize your plots with labels, titles, colors, and styles.

```
# Plot the sepal length and width variables
plt.scatter(df['SepalLengthCm'], df['SepalWidthCm'],
            c=df['SepalLengthCm'], cmap='rainbow',
           s=df['SepalWidthCm'] * 15, alpha=0.5, marker='*')
# Label the x and y axes
plt.xlabel('SepalLengthCm', fontname='Bookman Old Style', size=14,
color='blue')
plt.ylabel('SepalWidthCm', fontname='Bookman Old Style', size=14,
color='green')
# Give a title to the plot
plt.title('Scatter Plot of Sepal Length and Width of Iris Flowers', c='red')
# To show the color scale
plt.colorbar()
# Display the plot
plt.grid(True)
plt.show()
```

Output:

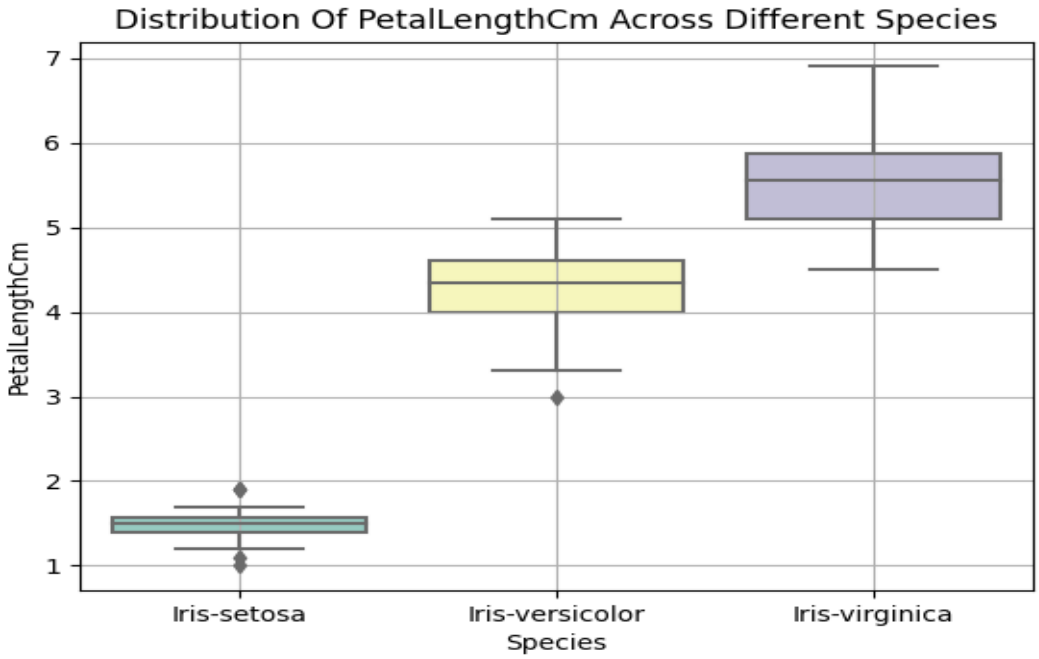


3. Seaborn Library

- Create a box plot to visualize the distribution of a numerical variable across different categories.
- Customize the appearance of seaborn plots using various parameters.

```
import seaborn as sns
import matplotlib.pyplot as plt
from Pandas import df
# Let's create a box plot for 'PetalLengthCm' for each species.
sns.boxplot(x='Species', y='PetalLengthCm', data=df, palette='Set3')
plt.xlabel('Species')
plt.ylabel('PetalLengthCm')
plt.title('Distribution Of PetalLengthCm Across Different Species')
plt.grid(True)
plt.show()
```

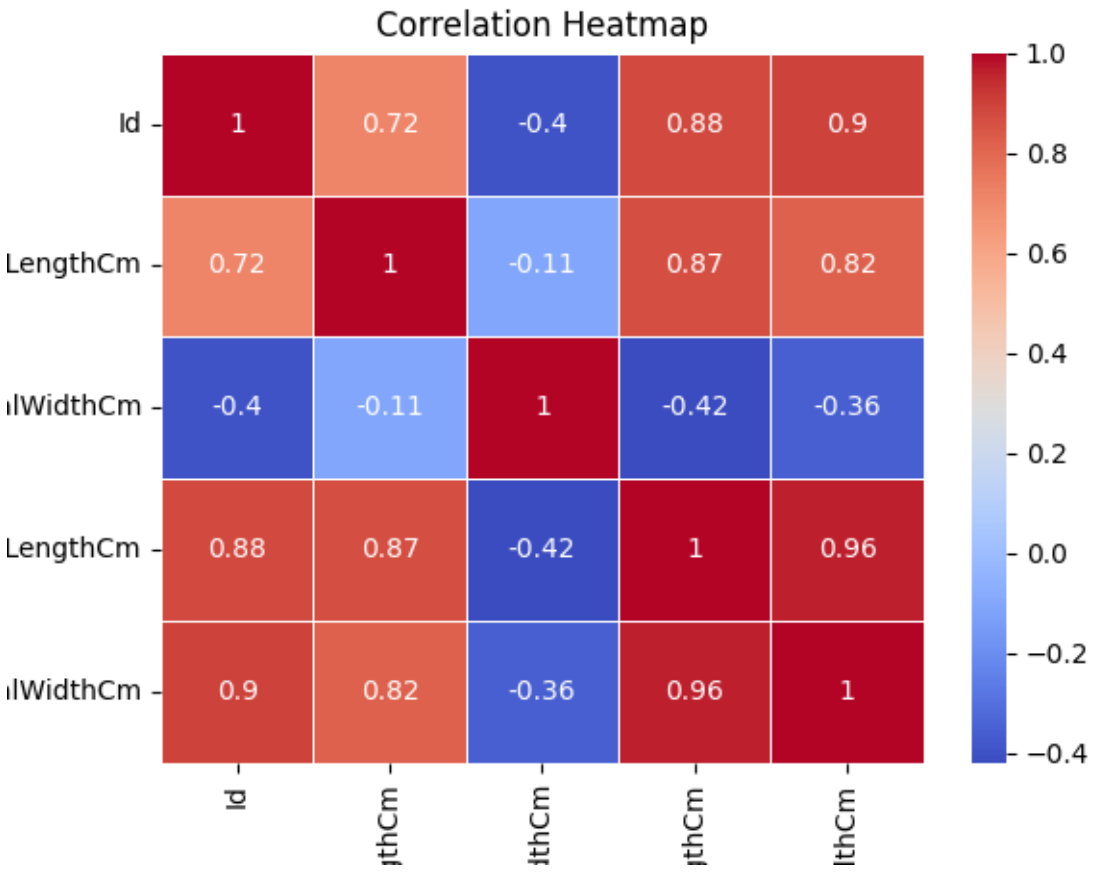
Output:-



- Generate a heatmap to explore the correlation between numerical variables.

```
# Let's plot the correlation heatmap for the numerical columns in the
dataset
numerical_column = df.select_dtypes(include=['float64', 'int64'])
sns.heatmap(numerical_column.corr(), annot=True, cmap='coolwarm',
linewidths=0.5)
plt.title('Correlation Heatmap')
plt.show()
```

Output:-



4. NumPy Library

- Create a NumPy array and perform basic operations like addition, subtraction, and multiplication.

```
import numpy as np
# Create a numpy array
arr1 = np.array([1, 2, 3, 4, 5])
arr2 = np.array([6, 7, 8, 9, 10])
# Basic operation
print("Addition Result: ", arr1 + arr2)
print("Subtraction Result: ", arr1 - arr2)
print("Multiplication Result: ", arr1 * arr2)
```

Output:-

```
Addition Result:  [ 7  9 11 13 15]
Subtraction Result:  [-5 -5 -5 -5 -5]
Multiplication Result:  [ 6 14 24 36 50]
```

- Use NumPy functions to calculate statistical measures like mean, median, and standard deviation.

```
# Statistical Measures
print("Mean of arr1: ", np.mean(arr1))
print("Median of arr2: ", np.median(arr2))
print("Standard Deviation of arr1: ", np.std(arr1))
```

Output:-

```
Mean of arr1:  3.0
Median of arr2:  8.0
Standard Deviation of arr1:  1.4142135623730951
```

- Reshape and slice NumPy arrays to extract specific data elements.

```
arr = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12])
reshaped_arr = arr.reshape(4, 3)
print("Reshaped Array is:-\n", reshaped_arr)
sliced_arr = arr[slice(-4, None)]
print("Sliced Array is:-\n", sliced_arr)
```

Output:-

```
Reshaped Array is:-
 [[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [10 11 12]]
Sliced Array is:-
 [ 9 10 11 12]
```

- Perform element-wise operations and broadcasting with NumPy arrays.

```
arr3 = np.array([[1, 2, 3, 4], [5, 10, 12, 15]])
arr4 = np.array([[6, 7, 8, 9], [10, 1, 7, 5]])
c = 2
print("Element wise addition:-\n", arr3 + arr4)
print("Element wise multiplication with Broadcasting:-\n", c * arr3)
```

Output:-

```
Element wise addition:-
 [[ 7  9 11 13]
 [15 11 19 20]]
Element wise multiplication with Broadcasting:-
 [[ 2  4  6  8]
 [10 20 24 30]]
```

- Apply mathematical functions (e.g., exponential, logarithm) to NumPy arrays.

```
arr5 = np.array([1, 2, 3, 4, 5])
print("Exponetial of arr5: ", np.exp(arr5))
print("Logarithim of arr5: ", np.log(arr5))
print("Square root of arr5: ", np.sqrt(arr5))
```

Output:-

```
Exponetial of arr5:  [  2.71828183   7.3890561   20.08553692  54.59815003 148.4131591 ]
Logarithim of arr5:  [0.         0.69314718 1.09861229 1.38629436 1.60943791]
Square root of arr5:  [1.         1.41421356 1.73205081 2.         2.23606798]
```

5. SciPy Library

- Use SciPy to perform numerical integration for a given mathematical function.

```
import scipy.integrate as integrate
# Define the values of a and b
a = 2
b = 3
# Define the integrand as a lambda expression
f = lambda x, y: a * x ** 2 + b
# Define the ranges of integration as a list of tuples
ranges = [(0, 1), (0, 1)]
# Call nquad with the integrand and the ranges
I, err = integrate.nquad(f, ranges)
# Print the result and the error estimate
print(f"Numerical integration result: {I:.4f}")
print(f"Prime estimated error: {err:.4e}")
```

Output:-

```
Numerical integration result: 3.6667
Prime estimated error: 4.0708e-14
```

Submitted By,
Name- Shankar Singh Mahanty
Regd. No.- 2101020758
Roll No.- CSE21238
Group- 3
Sem- 5th
Branch- CSE