# BIRLA INSTITUTE OF TECHNOLOGY

NOIDA OFF CAMPUS,MESRA


## PRACTICAL FILE
## MACHINE LEARNING

NAME:- Yunika Chadha
ROLL NO. :- BCA/45013/23
Department :- Computer Science [CSE]

# VISION

The department strives to be recognized for outstanding education and research, leading to excellent professionals and innovators in the field of Computer Science and Engineering, who can positively contribute to the society.

# MISION

The mission of the department is to impart quality education and research, to produce innovative and globally competent technocrats as well as young researchers having sound knowledge in the fields of Computer Science & Engineering.

1. To impart quality education and equip the students with a strong foundation that could make them capable of handling challenges of the new century.
2. To maintain state of the art research facilities and facilitate interaction with world's leading universities, industries and research organisations for constant improvement in the quality of education and research.

# INDEX

# PYTHON BASIC QUESTIONS

Q1. Check whether a number is even or odd.

```
num = int(input("Enter a number: "))

if num % 2 == 0:
    print(num, "is Even")
else:
    print(num, "is Odd")
```

```
Enter a number: 24351
 24351 is Odd
```

*Q2.* Find the largest among three numbers.

```
a, b, c = 12, 25, 9

if a >= b and a >= c:
    print("Largest:", a)
elif b >= a and b >= c:
    print("Largest:", b)
else:
    print("Largest:", c)
```

```
Largest: 25
```

Q3. Display multiplication table of a number.

```
n = 5
print("Multiplication Table of", n)
for i in range(1, 11):
    print(n, "x", i, "=", n*i)
```

```
Multiplication Table of 5
 5 x 1 = 5
 5 x 2 = 10
 5 x 3 = 15
 5 x 4 = 20
 5 x 5 = 25
 5 x 6 = 30
 5 x 7 = 35
 5 x 8 = 40
 5 x 9 = 45
 5 x 10 = 50
```

Q4. Find factorial using a for loop.

```
n = 5
fact = 1
for i in range(1, n+1):
    fact *= i
print("Factorial of", n, "=", fact)
```

```
Factorial of 5 = 120
```

Q5. Find sum of digits of a number.

```
num = 1234
s = 0
while num > 0:
    digit = num % 10
    s += digit
    num //= 10
print("Sum of digits =", s)
```

```
Sum of digits = 10
```

Q6. Reverse a number using while loop.

```
num = 12345
rev = 0
while num > 0:
    digit = num % 10
    rev = rev*10 + digit
    num //= 10
print("Reversed number =", rev)
```

```
Reversed number = 54321
```

Q7. Check if a string is a palindrome.

```
s = "madam"
if s == s[::-1]:
    print(s, "is a Palindrome")
else:
    print(s, "is Not a Palindrome")
```

```
madam is a Palindrome
```

Q8. Count vowels in a string.

```
s = "Python Programming"
vowels = "aeiouAEIOU"
count = 0
for ch in s:
    if ch in vowels:
```

```
        count += 1
print("Number of vowels =", count)
```

```
Number of vowels = 4
```

Q9. Perform operations on list.

```
nums = [10, 25, 7, 50, 32]
print("List:", nums)
print("Largest:", max(nums))
print("Smallest:", min(nums))
print("Sum:", sum(nums))
```

```
List: [10, 25, 7, 50, 32]
Largest: 50
Smallest: 7
Sum: 124
```

Q10. Work with dictionary of students.

```
students = {"Aman":85, "Riya":92, "Kabir":76, "Neha":89, "Sam":95}

print("Students and Marks:", students)

highest = max(students, key=students.get)
print("Topper:", highest, "with", students[highest], "marks")
```

```
Students and Marks: {'Aman': 85, 'Riya': 92, 'Kabir': 76, 'Neha': 89, 'Sam': 95}
Topper: Sam with 95 marks
```

## PANDAS BASIC QUESTIONS

Q1. Load dataset and display first rows

```
from sklearn.datasets import load_iris
import pandas as pd

iris = load_iris(as_frame=True)
df = iris.frame
print("Shape:", df.shape)
print("Columns:", df.columns.tolist())
print(df.head())
```

```
Shape: (150, 5)
Columns: ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
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

   target
0       0
1       0
2       0
3       0
4       0
```

Q2. Filter rows and show specific columns

```
filtered = df[df['sepal length (cm)'] > 5.0]
print("Filtered Shape:", filtered.shape)
print(filtered[['sepal length (cm)', 'target']].head())
```

```
Filtered Shape: (118, 5)
    sepal length (cm)  target
0                 5.1       0
5                 5.4       0
10                5.4       0
14                5.8       0
15                5.7       0
```

Q3. Group by target and calculate means

```
grouped = df.groupby("target").mean()
print("Mean values by species:")
print(grouped)
```

```
Mean values by species:
        sepal length (cm)  sepal width (cm)  petal length (cm)  \
target
0                   5.006             3.428              1.462
1                   5.936             2.770              4.260
2                   6.588             2.974              5.552

        petal width (cm)
target
0                  0.246
1                  1.326
2                  2.026
```

Q4. Handle missing values

```
df2 = df.copy()
df2.loc[0, 'sepal length (cm)'] = None   # introduce NaN
print("Before:", df2.isna().sum())
df2['sepal length (cm)'].fillna(df2['sepal length (cm)'].mean(), inplace=True)
print("After:", df2.isna().sum())
```

```
Before: sepal length (cm)    1
sepal width (cm)     0
```

```
petal length (cm)    0
petal width (cm)     0
target               0
dtype: int64
After: sepal length (cm)    0
sepal width (cm)     0
petal length (cm)    0
petal width (cm)     0
target               0
dtype: int64
/tmp/ipython-input-2689347202.py:4: FutureWarning: A value is trying to be set on a c
The behavior will change in pandas 3.0. This inplace method will never work because t

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({


  df2['sepal length (cm)'].fillna(df2['sepal length (cm)'].mean(), inplace=True)
```

### Q5. New column and top 3 values

```
df['petal_area'] = df['petal length (cm)'] * df['petal width (cm)']
top3 = df.sort_values(by='petal_area', ascending=False).head(3)
print("Top 3 rows with largest petal area:")
print(top3[['petal length (cm)', 'petal width (cm)', 'petal_area']])
```

```
Top 3 rows with largest petal area:
     petal length (cm)  petal width (cm)  petal_area
118                6.9               2.3       15.87
109                6.1               2.5       15.25
100                6.0               2.5       15.00
```

## NUMPY BASIC QUESTIONS

### Q1. Create array and extract evens

```
import numpy as np

arr = np.arange(1, 21)
print("Original:", arr)
print("Evens:", arr[arr % 2 == 0])
```

```
Original: [ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20]
Evens: [ 2  4  6  8 10 12 14 16 18 20]
```

### Q2. Identity × random matrix

```
I = np.eye(3)
A = np.random.randint(1, 10, (3, 3))
print("Matrix A:\n", A)
print("I x A:\n", I @ A)
```

```
Matrix A:
 [[7 2 8]
 [4 9 2]
 [4 1 5]]
I x A:
 [[7. 2. 8.]
 [4. 9. 2.]
 [4. 1. 5.]]
```

### Q3. Reshape and flatten

```
arr = np.arange(12)
print("Original:", arr)
mat = arr.reshape(3,4)
print("Reshaped:\n", mat)
print("Flattened:", mat.flatten())
```

```
Original: [ 0  1  2  3  4  5  6  7  8  9 10 11]
Reshaped:
 [[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]
Flattened: [ 0  1  2  3  4  5  6  7  8  9 10 11]
```

### Q4. Normal distribution statistics

```
data = np.random.normal(0, 1, 100)
print("Mean:", round(data.mean(),2))
print("Variance:", round(data.var(),2))
print("Std Dev:", round(data.std(),2))
```

```
Mean: 0.08
Variance: 0.84
Std Dev: 0.92
```

### Q5. Normalize array

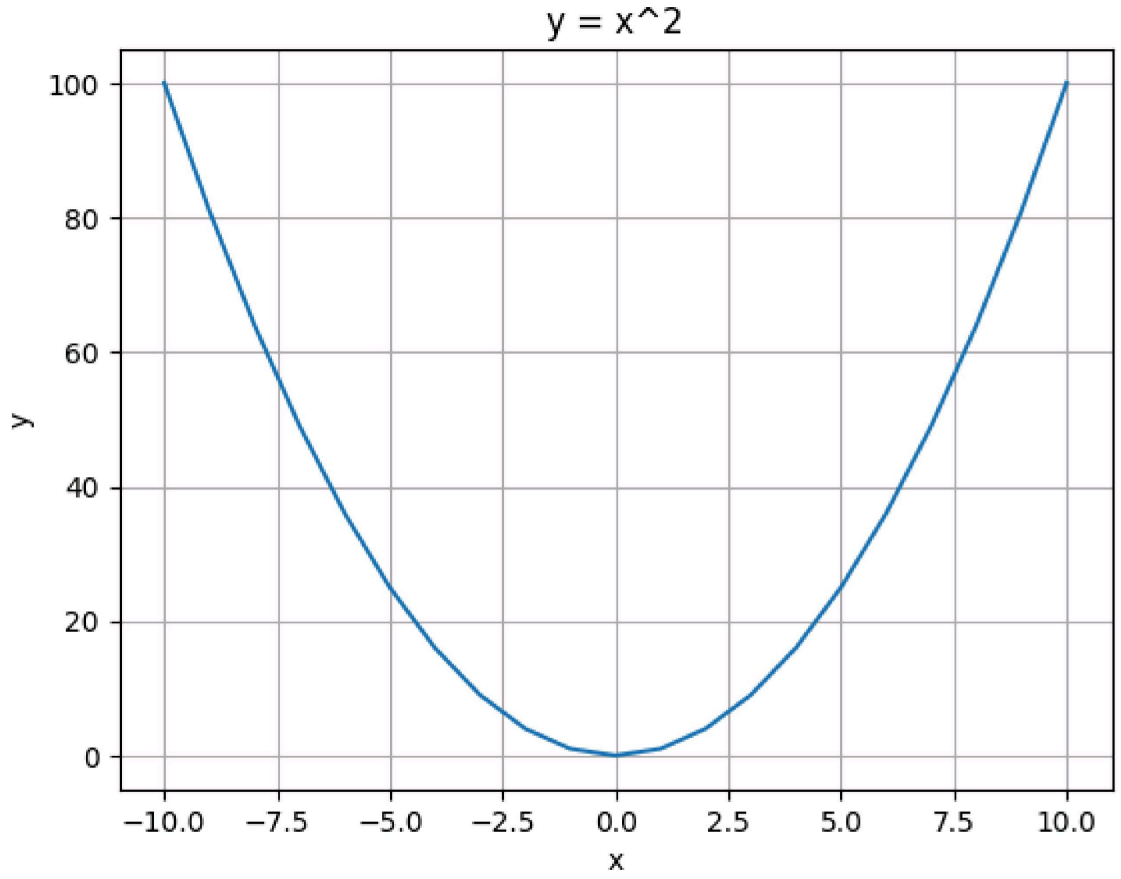
```
def normalize(x):
    return (x - x.min()) / (x.max() - x.min())

arr = np.array([10,20,30,40,50])
print("Original:", arr)
print("Normalized:", normalize(arr))
```

```
Original: [10 20 30 40 50]
Normalized: [0.   0.25 0.5  0.75 1.  ]
```

# MATPLOTLIB BASIC QUESTIONS

Q1. Plot $y = x^2$

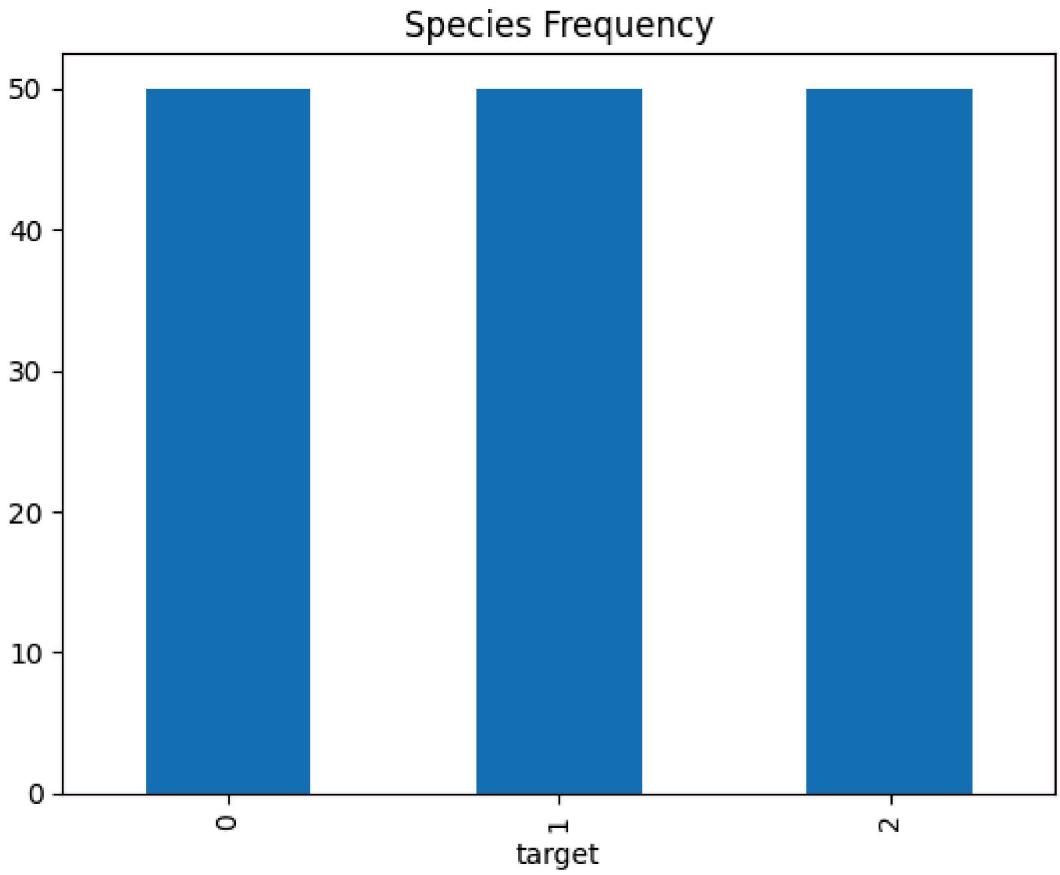
```
import matplotlib.pyplot as plt

x = np.arange(-10,11)
y = x**2
plt.plot(x,y)
plt.title("y = x^2")
plt.xlabel("x")
plt.ylabel("y")
plt.grid(True)
plt.show()
```




Q2. Bar chart of species frequency

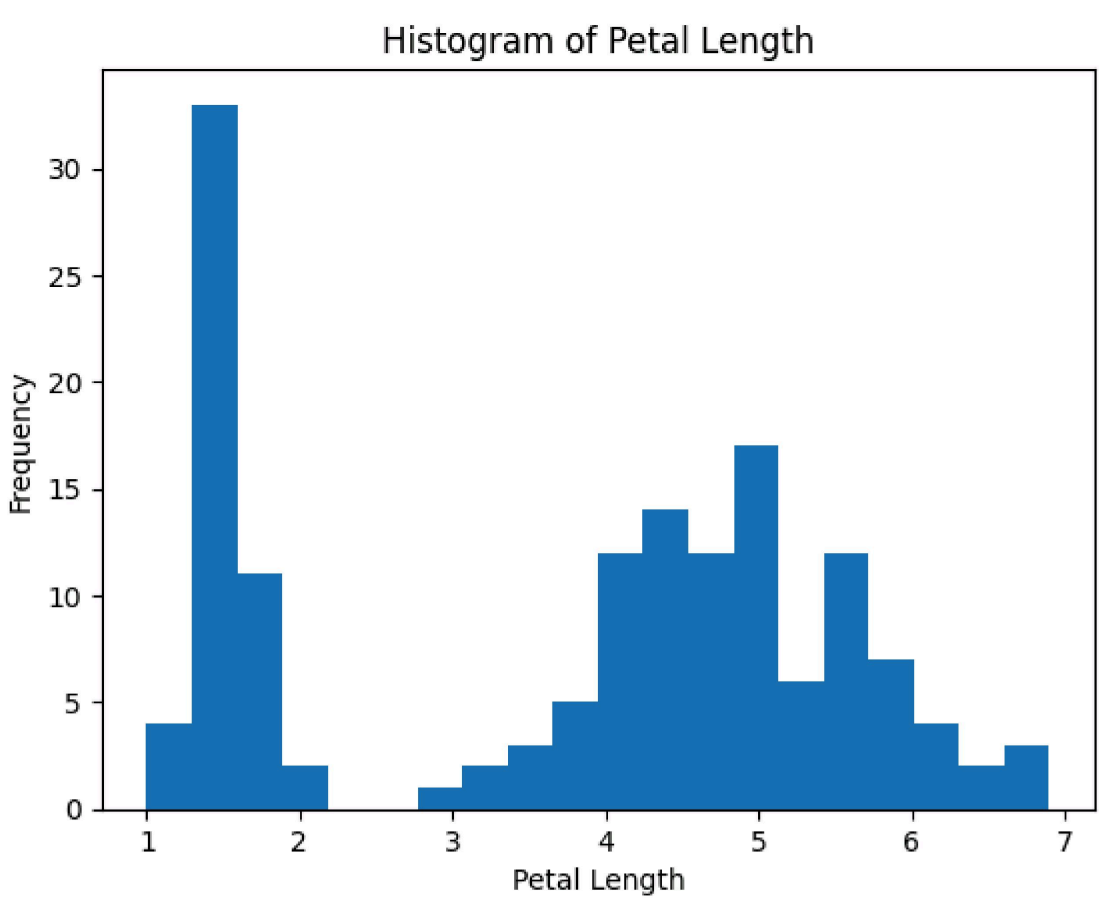
```
counts = df['target'].value_counts()
print("Species counts:\n", counts)
counts.plot(kind='bar')
plt.title("Species Frequency")
plt.show()
```

```
Species counts:
 target
0    50
1    50
2    50
Name: count, dtype: int64
```



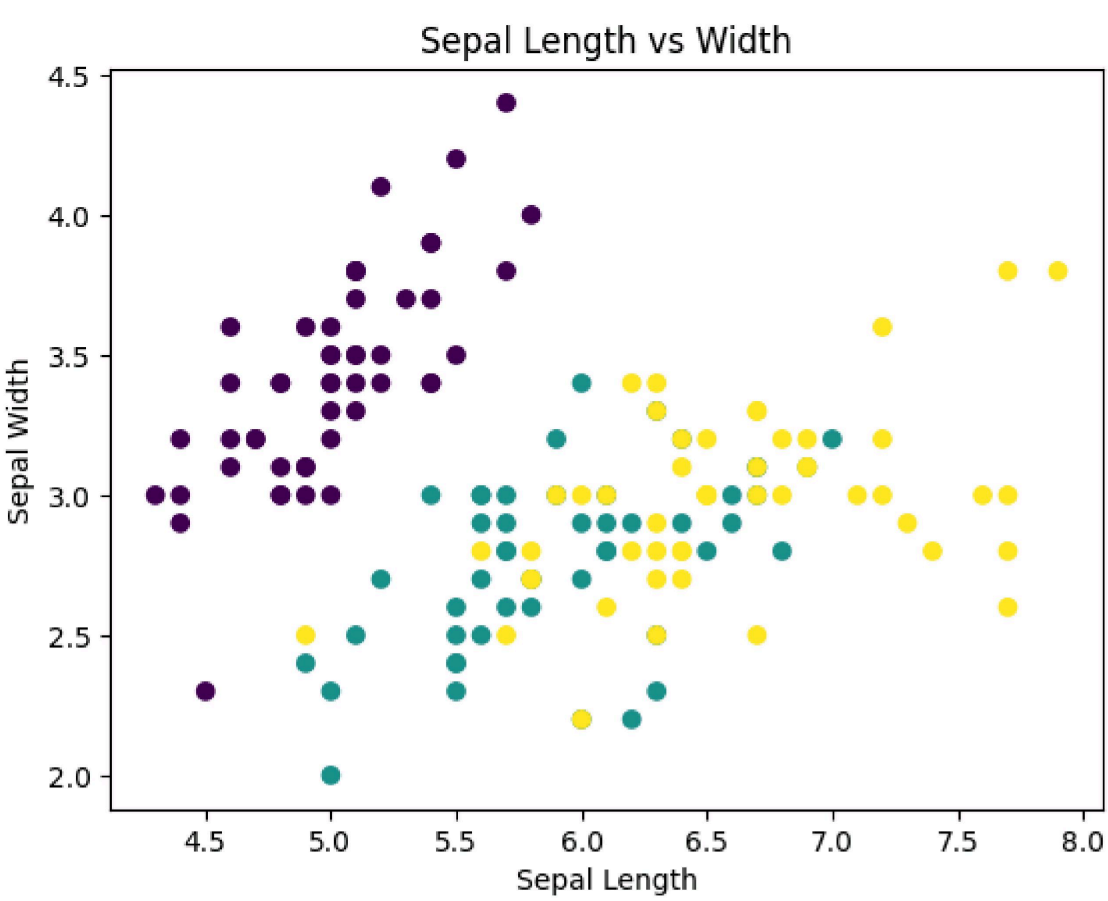

Q3. Histogram of petal length

```
plt.hist(df['petal length (cm)'], bins=20)
plt.title("Histogram of Petal Length")
plt.xlabel("Petal Length")
plt.ylabel("Frequency")
plt.show()
```



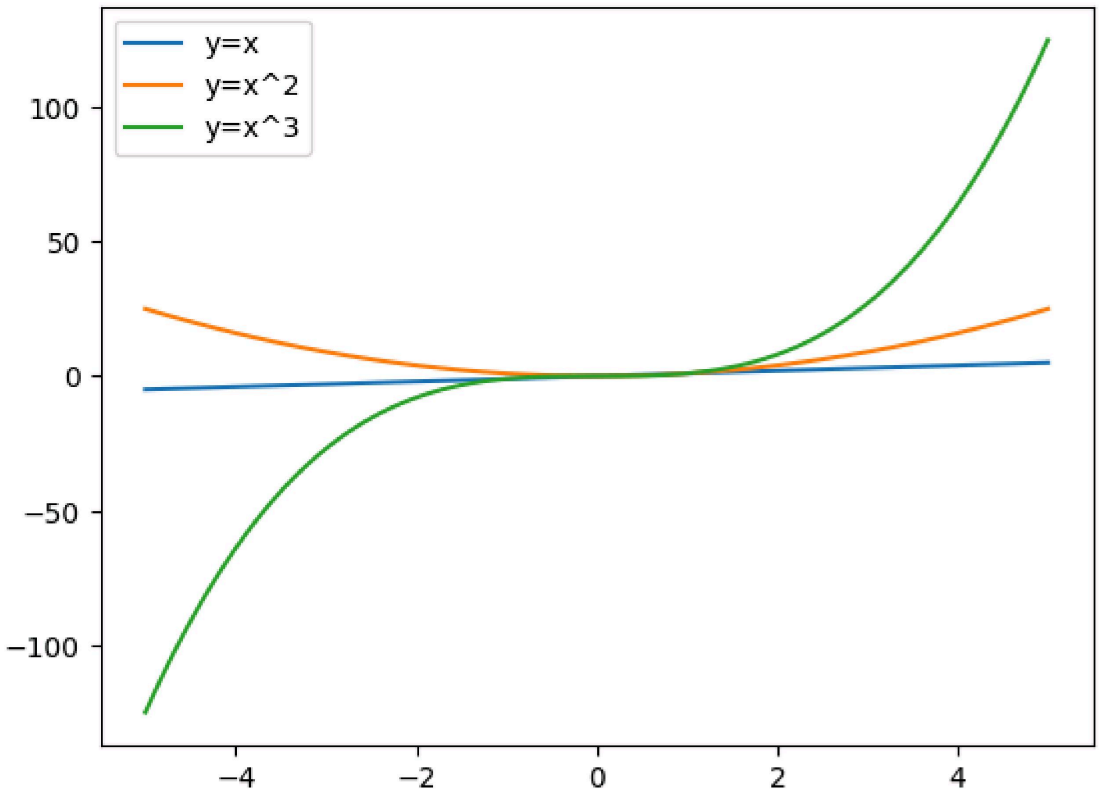

Q4. Scatter plot

```
plt.scatter(df['sepal length (cm)'], df['sepal width (cm)'], c=df['target'])
plt.xlabel("Sepal Length")
plt.ylabel("Sepal Width")
plt.title("Sepal Length vs Width")
plt.show()
```




Q5. Multiple lines

```
x = np.linspace(-5,5,100)
plt.plot(x,x,label="y=x")
plt.plot(x,x**2,label="y=x^2")
plt.plot(x,x**3,label="y=x^3")
plt.legend()
plt.show()
```




## SCIKIT BASIC QUESTIONS

Q1. Train-test split

```
from sklearn.model_selection import train_test_split

X = df.iloc[:,:4]
y = df['target']
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2,random_state=42)
print("Train shape:", X_train.shape)
print("Test shape:", X_test.shape)
```

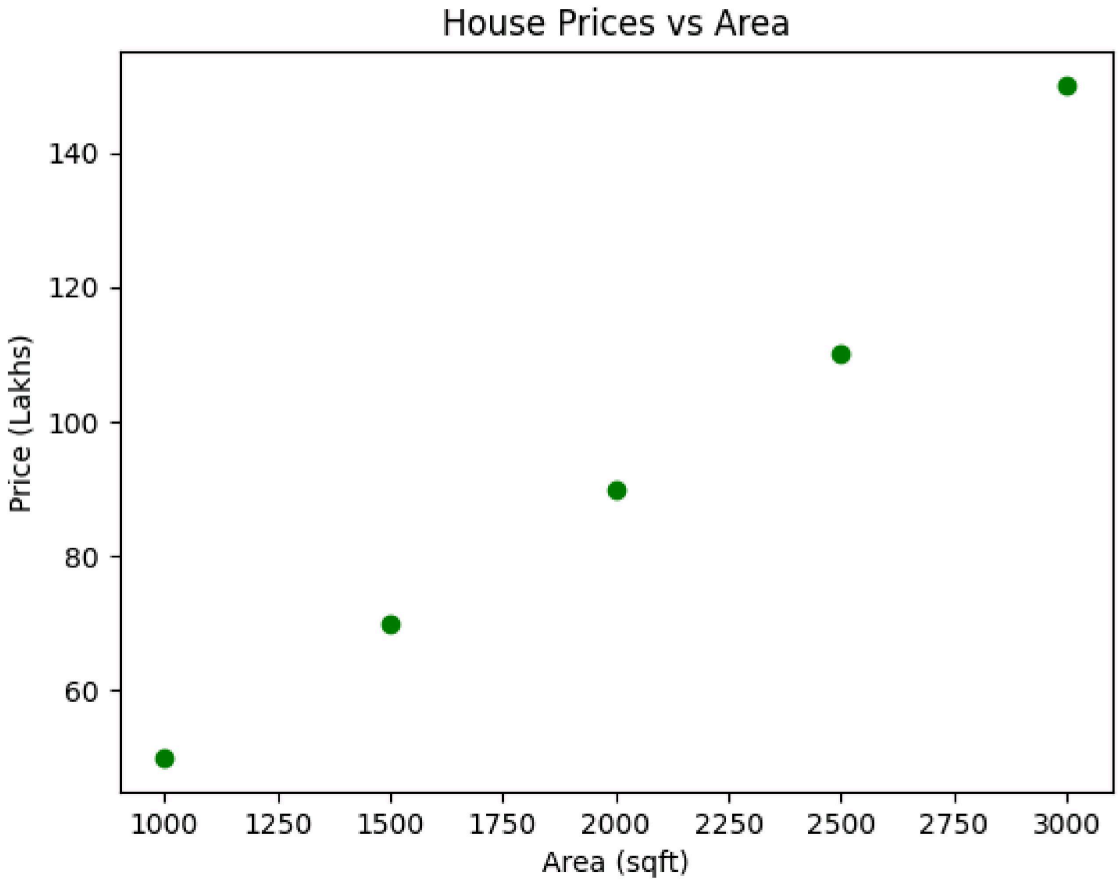
```
Train shape: (120, 4)
Test shape: (30, 4)
```

## 1. Introduction to the language – Importing datasets – Data visualization.

```
import pandas as pd
import matplotlib.pyplot as plt

data = {
    'area': [1000, 1500, 2000, 2500, 3000],
    'price': [50, 70, 90, 110, 150]
}
```

```
df = pd.DataFrame(data)
print(df.head())

plt.scatter(df['area'], df['price'], color='green')
plt.xlabel('Area (sqft)')
plt.ylabel('Price (Lakhs)')
plt.title('House Prices vs Area')
plt.show()
```

```
   area  price
0  1000     50
1  1500     70
2  2000     90
3  2500    110
4  3000    150
```




## 2.Implement Linear Regression problem. For example, based on a dataset comprising of existing set of prices and area/size of the houses, predict the estimated price of a given house.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

data = {
    'area': [1000, 1500, 2000, 2500, 3000],
```

```
    'price': [50, 70, 90, 110, 150]
}

df = pd.DataFrame(data)
X = df[['area']]
y = df['price']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = LinearRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
print("Predicted prices:", y_pred)
print("Actual prices:", y_test.values)

new_area = np.array([[2200]])
predicted_price = model.predict(new_area)
print("Predicted price for 2200 sqft:", predicted_price[0])

plt.scatter(df['area'], df['price'], color='blue', label='Data points')
plt.plot(df['area'], model.predict(df[['area']]), color='red', label='Regression line')
plt.xlabel('Area (sqft)')
plt.ylabel('Price (Lakhs)')
plt.legend()
plt.show()
```

## 3. Based on multiple features/variables perform Linear Regression. For example, based on a number of additional features like number of bedrooms, servant room, number of balconies, number of houses of years a house has been built – predict the price of a house. solve with code solution already done for 1 , do 2 and 3 as done by basic knowledge programmer of machine learning

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import numpy as np

data2 = {
    'area': [1000, 1500, 2000, 2500, 3000],
    'bedrooms': [2, 3, 3, 4, 4],
    'balconies': [1, 2, 2, 3, 3],
    'age': [5, 10, 15, 20, 25],
    'price': [50, 65, 85, 110, 150]
}

df2 = pd.DataFrame(data2)
X = df2[['area', 'bedrooms', 'balconies', 'age']]
y = df2['price']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model2 = LinearRegression()
model2.fit(X_train, y_train)

y_pred = model2.predict(X_test)
print("Predicted prices:", y_pred)
print("Actual prices:", y_test.values)

new_house = np.array([[2200, 3, 2, 12]])
predicted_price = model2.predict(new_house)
print("Predicted price for new house:", predicted_price[0])
```

```
Predicted prices: [60.]
Actual prices: [65]
Predicted price for new house: 108.99650034996513
/usr/local/lib/python3.12/dist-packages/sklearn/utils/validation.py:2739: UserWarning:
  warnings.warn(
```