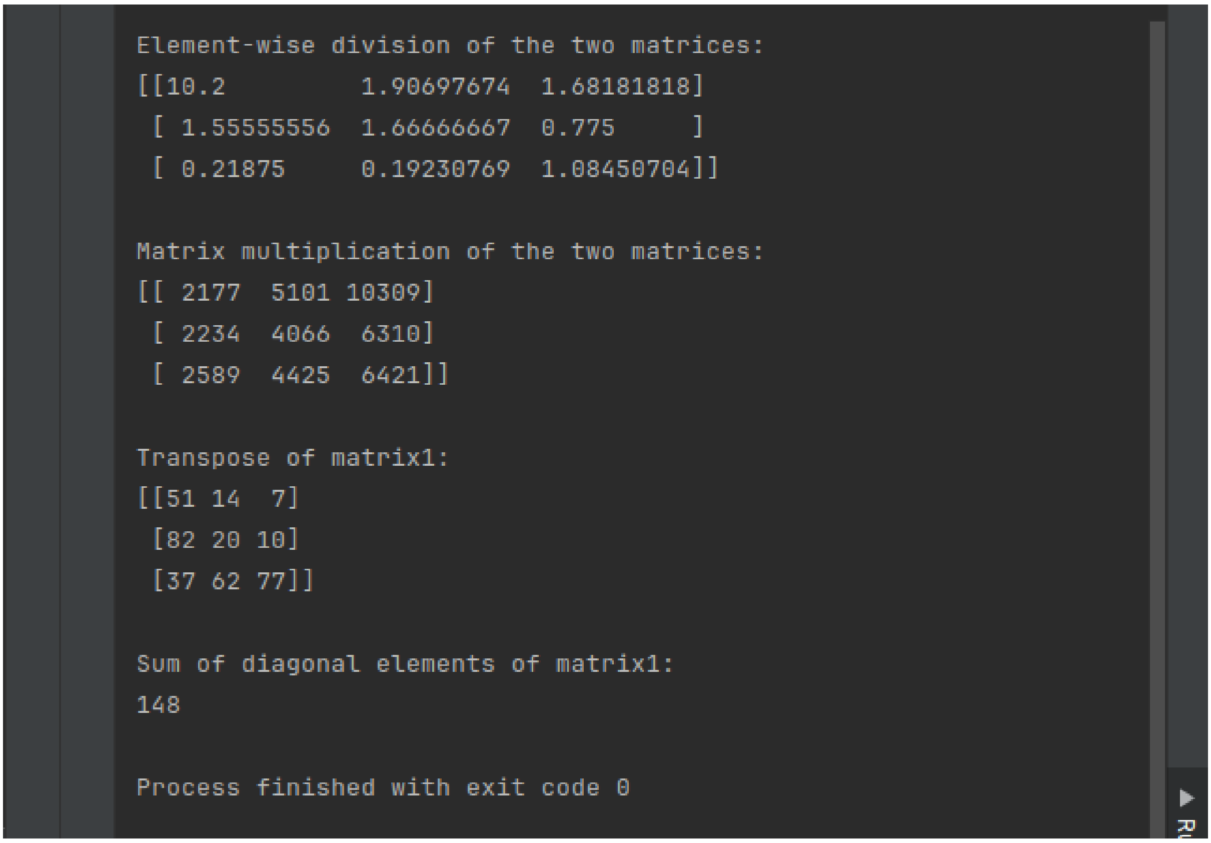
1. Write a program to perform different matrix operations on a 2D Matrix

Input:

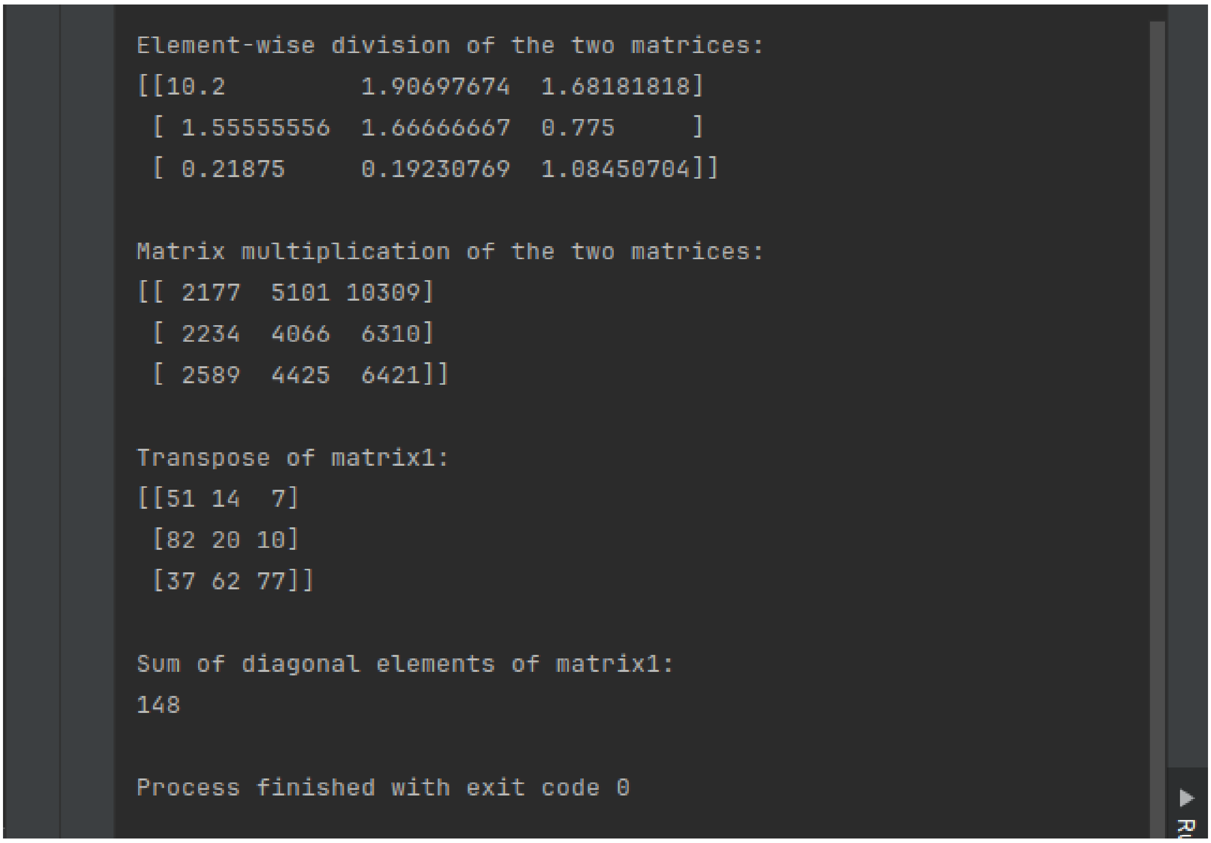
```
import numpy as np
print("Name: APARNA MOHAN")
print("Reg No: SJC22MCA-2013")
print("Batch: 22-24")
print()
matrix1 = np.array([[51, 82, 37], [14, 20, 62], [7, 10, 77]])
matrix2 = np.array([[5, 43, 22], [9, 12, 80], [32, 52, 71]])
print("Matrix 1:")
print(matrix1)
print()
print("Matrix 2:")
print(matrix2)
print()
matrix_sum = matrix1 + matrix2
print("Sum of the two matrices:")
print(matrix_sum)

matrix_diff = matrix1 - matrix2
print("\nDifference of the two matrices:")
print(matrix_diff)

matrix_product = matrix1 * matrix2
print("\nElement-wise product of the two matrices:")
print(matrix_product)
```

```
with np.errstate(divide='ignore', invalid='ignore'):
    matrix_division = np.true_divide(matrix1, matrix2)
    matrix_division[~np.isfinite(matrix_division)] = np.nan
print("\nElement-wise division of the two matrices:")
print(matrix_division)

matrix_mult = np.dot(matrix1, matrix2)
print("\nMatrix multiplication of the two matrices:")
print(matrix_mult)



matrix1_transpose = np.transpose(matrix1)
print("\nTranspose of matrix1:")
print(matrix1_transpose)

diagonal_sum = np.trace(matrix1)
print("\nSum of diagonal elements of matrix1:")
print(diagonal_sum)
```

Output:



```
Element-wise division of the two matrices:
[[10.2         1.90697674  1.68181818]
 [ 1.55555556  1.66666667  0.775     ]
 [ 0.21875     0.19230769  1.08450704]]

Matrix multiplication of the two matrices:
[[ 2177  5101 10309]
 [ 2234  4066  6310]
 [ 2589  4425  6421]]

Transpose of matrix1:
[[51 14  7]
 [82 20 10]
 [37 62 77]]

Sum of diagonal elements of matrix1:
148

Process finished with exit code 0
```



```
Element-wise division of the two matrices:
[[10.2         1.90697674  1.68181818]
 [ 1.55555556  1.66666667  0.775     ]
 [ 0.21875     0.19230769  1.08450704]]

Matrix multiplication of the two matrices:
[[ 2177  5101 10309]
 [ 2234  4066  6310]
 [ 2589  4425  6421]]

Transpose of matrix1:
[[51 14  7]
 [82 20 10]
 [37 62 77]]

Sum of diagonal elements of matrix1:
148

Process finished with exit code 0
```

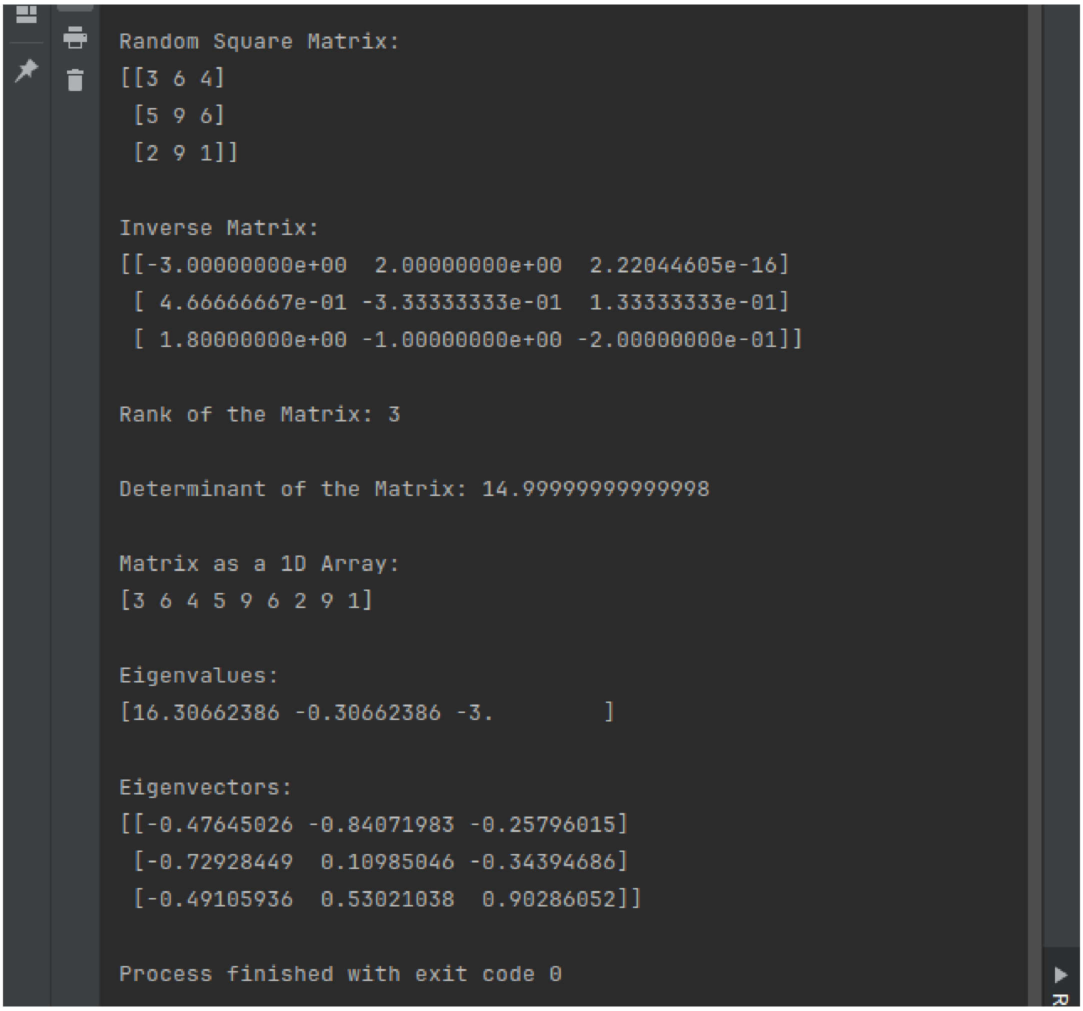
2. Write a program to find the inverse, rank, determinant, Eigen values of a given matrix. Also transform the matrix to 1D array.

Input:

```
import numpy as np
print("Name: APARNA MOHAN")
print("Reg No: SJC22MCA-2013")
print("Batch: 22-24")
print()
matrix_size = 3

random_matrix = np.random.randint(1, 11, size=(matrix_size, matrix_size))

print("Random Square Matrix:")
print(random_matrix)

try:
    inverse_matrix = np.linalg.inv(random_matrix)
    print("\nInverse Matrix:")
    print(inverse_matrix)
except np.linalg.LinAlgError:
    print("\nInverse does not exist for this matrix.")

rank = np.linalg.matrix_rank(random_matrix)
print("\nRank of the Matrix:", rank)

determinant = np.linalg.det(random_matrix)
print("\nDeterminant of the Matrix:", determinant)
```

```
matrix_1d = random_matrix.flatten()

print("\nMatrix as a 1D Array:")

print(matrix_1d)

eigenvalues, eigenvectors = np.linalg.eig(random_matrix)

print("\nEigenvalues:")

print(eigenvalues)

print("\nEigenvectors:")

print(eigenvectors)
```

Output:



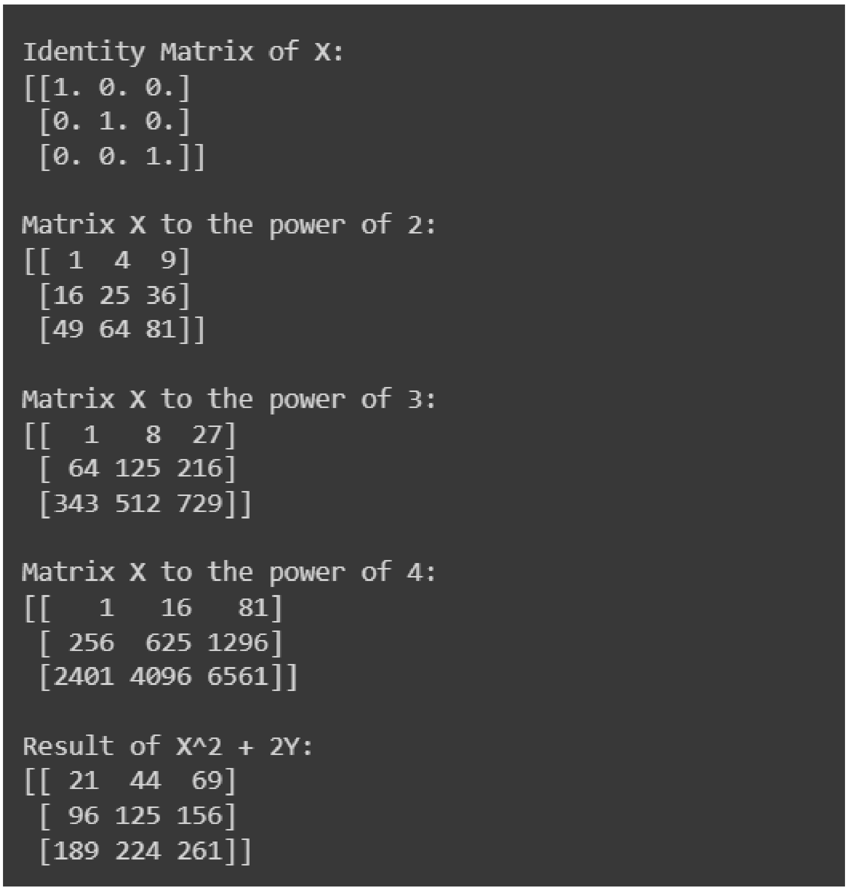
```
Random Square Matrix:
[[3 6 4]
 [5 9 6]
 [2 9 1]]

Inverse Matrix:
[[-3.00000000e+00  2.00000000e+00  2.22044605e-16]
 [ 4.66666667e-01 -3.33333333e-01  1.33333333e-01]
 [ 1.80000000e+00 -1.00000000e+00 -2.00000000e-01]]

Rank of the Matrix: 3

Determinant of the Matrix: 14.99999999999998

Matrix as a 1D Array:
[3 6 4 5 9 6 2 9 1]

Eigenvalues:
[16.30662386 -0.30662386 -3.        ]

Eigenvectors:
[[-0.47645026 -0.84071983 -0.25796015]
 [-0.72928449  0.10985046 -0.34394686]
 [-0.49105936  0.53021038  0.90286052]]

Process finished with exit code 0
```

3. Write a program to display the elements of the matrix X to different powers and identity matrix of a given matrix .Also create another matrix Y with same dimensions and display X^2+2Y

Input:

```
import numpy as np

print("Name: : APARNA MOHAN ")

print("Reg No: SJC22MCA-2013")

print("Batch: 22-24")

print()

X = np.array([[1, 2, 3],

              [4, 5, 6],

              [7, 8, 9]])


identity_matrix = np.identity(X.shape[0])

print("\nIdentity Matrix of X:")

print(identity_matrix)

exponentials = [2, 3, 4]

powered_matrices = [np.power(X, exp) for exp in exponentials]

for i, exp in enumerate(exponentials):

    print(f"\nMatrix X to the power of {exp}:")

    print(powered_matrices[i])

Y = np.array([[10, 20, 30],

              [40, 50, 60],

              [70, 80, 90]])

result = np.power(X, 2) + 2 * Y

print("\nResult of X^2 + 2Y:")

print(result)
```

Output:



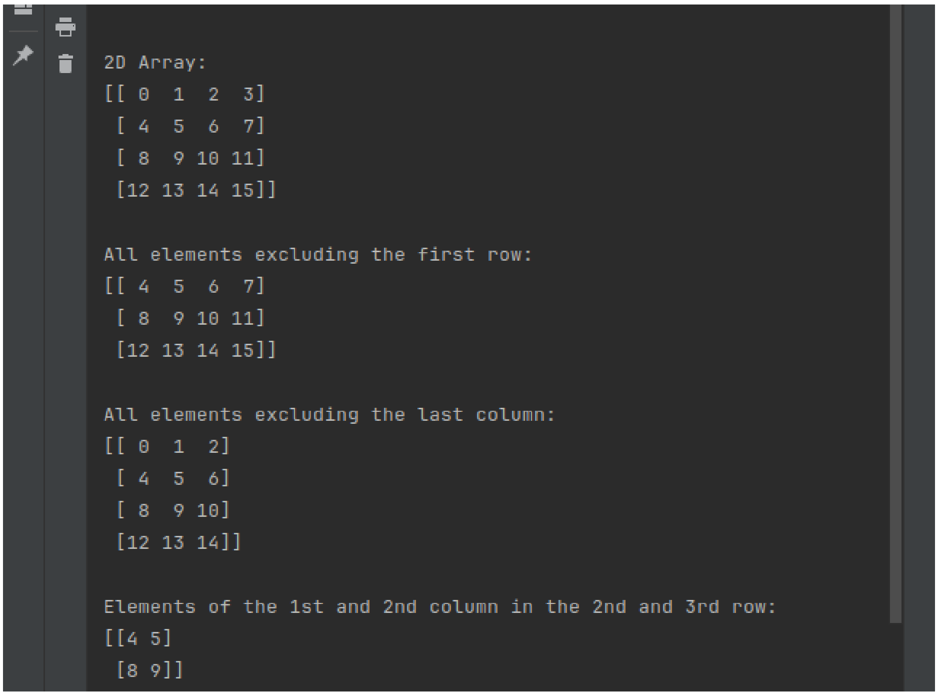
```
Identity Matrix of X:
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]

Matrix X to the power of 2:
[[ 1  4  9]
 [16 25 36]
 [49 64 81]]

Matrix X to the power of 3:
[[  1   8  27]
 [ 64 125 216]
 [343 512 729]]

Matrix X to the power of 4:
[[   1   16   81]
 [ 256  625 1296]
 [2401 4096 6561]]

Result of X^2 + 2Y:
[[ 21  44  69]
 [ 96 125 156]
 [189 224 261]]
```

4. Write a Program to display various elements of a give 4x4 matrix specifying appropriate indices

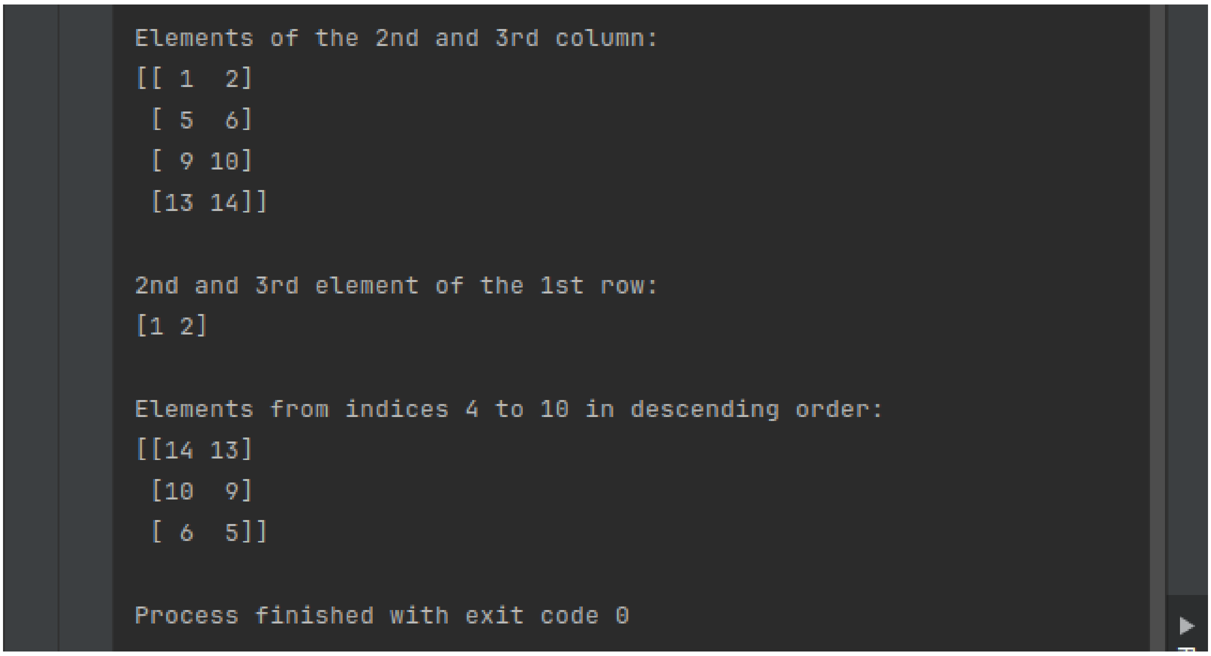
Input:

```
import numpy as np
print("Name: : APARNA MOHAN ")
print("Reg No: SJC22MCA-2013")
print("Batch: 22-24")
print()
two_dim_array = np.arange(16).reshape(4, 4)
print("\n2D Array:")
print(two_dim_array)

print("\nAll elements excluding the first row:")
print(two_dim_array[1:])

print("\nAll elements excluding the last column:")
print(two_dim_array[:, :-1])

print("\nElements of the 1st and 2nd column in the 2nd and 3rd row:")
print(two_dim_array[1:3, 0:2])

print("\nElements of the 2nd and 3rd column:")
print(two_dim_array[:, 1:3])

print("\n2nd and 3rd element of the 1st row:")
print(two_dim_array[0, 1:3])
print("\nElements from indices 4 to 10 in descending order:")

print(two_dim_array[3:0:-1, 2:0:-1])
```

Output:



```
2D Array:
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]]

All elements excluding the first row:
[[ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]]

All elements excluding the last column:
[[ 0  1  2]
 [ 4  5  6]
 [ 8  9 10]
 [12 13 14]]

Elements of the 1st and 2nd column in the 2nd and 3rd row:
[[4 5]
 [8 9]]
```



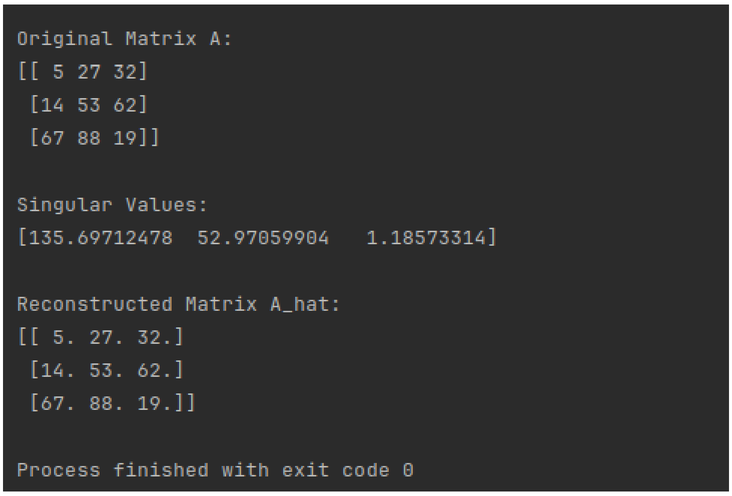
```
Elements of the 2nd and 3rd column:
[[ 1  2]
 [ 5  6]
 [ 9 10]
 [13 14]]

2nd and 3rd element of the 1st row:
[1 2]

Elements from indices 4 to 10 in descending order:
[[14 13]
 [10  9]
 [ 6  5]]

Process finished with exit code 0
```

5. Write a program to perform the SVD of a given matrix.

Input:

```
import numpy as np

print("Name: : APARNA MOHAN ")

print("Reg No: SJC22MCA-2013")

print("Batch: 22-24")

print()

A = np.array([[5, 27, 32], [14, 53, 62], [67, 88, 19]])

U, S, Vt = np.linalg.svd(A)

A_hat = U @ np.diag(S) @ Vt

print("Original Matrix A:")

print(A)

print("\nSingular Values:")

print(S)

print("\nReconstructed Matrix A_hat:")

print(A_hat)
```

Output:



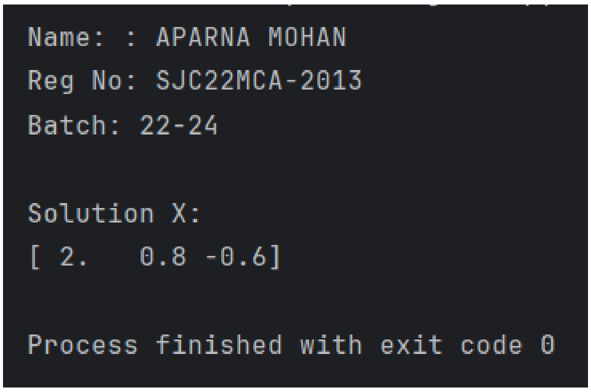
```
Original Matrix A:
[[ 5 27 32]
 [14 53 62]
 [67 88 19]]

Singular Values:
[135.69712478  52.97059904   1.18573314]

Reconstructed Matrix A_hat:
[[ 5. 27. 32.]
 [14. 53. 62.]
 [67. 88. 19.]]

Process finished with exit code 0
```

6. Write a program to Solve systems of equations with numpy

Input:

```
import numpy as np

print("Name: : APARNA MOHAN ")

print("Reg No: SJC22MCA-2013")

print("Batch: 22-24")

print()

A = np.array([[2, 3, -1],

              [1, 2, 1],

              [3, 1, -2]])

b = np.array([7, 3, 8])

try:

    X = np.linalg.solve(A, b)

    print("Solution X:")

    print(X)

except np.linalg.LinAlgError:

    print("Matrix A is singular. The system of equations may not have a unique
solution.")
```

Output:



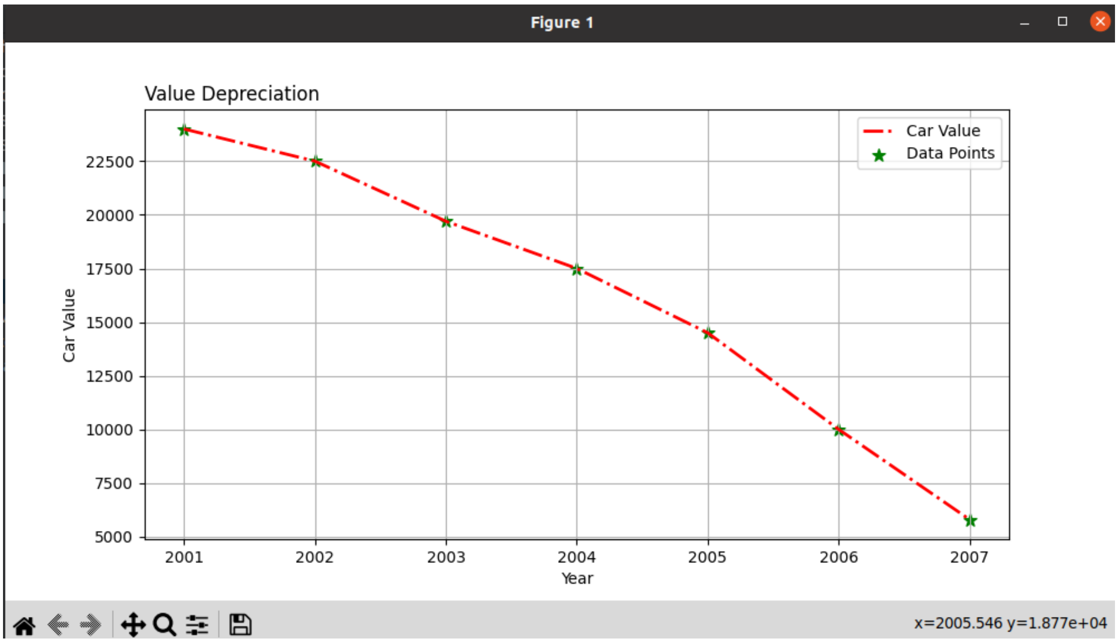
```
Name: : APARNA MOHAN
Reg No: SJC22MCA-2013
Batch: 22-24

Solution X:
[ 2.   0.8 -0.6]

Process finished with exit code 0
```

7. Program to create a line graph with the specified style properties, given the information regarding the car details.

Input:

```
import matplotlib.pyplot as plt


years = [2001, 2002, 2003, 2004, 2005, 2006, 2007]

car_values = [24000, 22500, 19700, 17500, 14500, 10000, 5800]

plt.figure(figsize=(10, 5))

plt.subplot(1, 1, 1)

plt.plot(years, car_values, 'r-.', label='Car Value', linewidth=2)

plt.scatter(years, car_values, c='green', marker='*', s=70, label='Data Points')

plt.xlabel('Year')

plt.ylabel('Car Value')

plt.title('Value Depreciation', loc='left')

plt.legend()

plt.grid(True)

plt.show()
```

Output:



8. Program to represent the daily sales of the 2 items in a shop using line graph with grids and appropriate style properties.

Input:

```
import matplotlib.pyplot as plt

days = ['Mon', 'Tues', 'Wed', 'Thurs', 'Fri']

sales_drinks = [300, 450, 150, 400, 650]

sales_food = [400, 500, 350, 300, 500]

fig, axes = plt.subplots(2, 1, figsize=(8, 10))

axes[0].plot(days, sales_drinks, linestyle='dotted', color='cyan', marker='H', markersize=8, markerfacecolor='magenta', markeredgecolor='black')

axes[0].set_xlabel('Days of week')

axes[0].set_ylabel('Sale of Drinks')

axes[0].set_title('Sales Data1', loc='right')

axes[0].grid(True, color='blue')

axes[1].plot(days, sales_food, linestyle='dashed', color='yellow', marker='D', markersize=8, markerfacecolor='green', markeredgecolor='red')

axes[1].set_xlabel('Days of week')

axes[1].set_ylabel('Sale of Food')

axes[1].set_title('Sales Data2', loc='center')

axes[1].grid(True, color='blue')

plt.tight_layout()

plt.show()
```

Output:

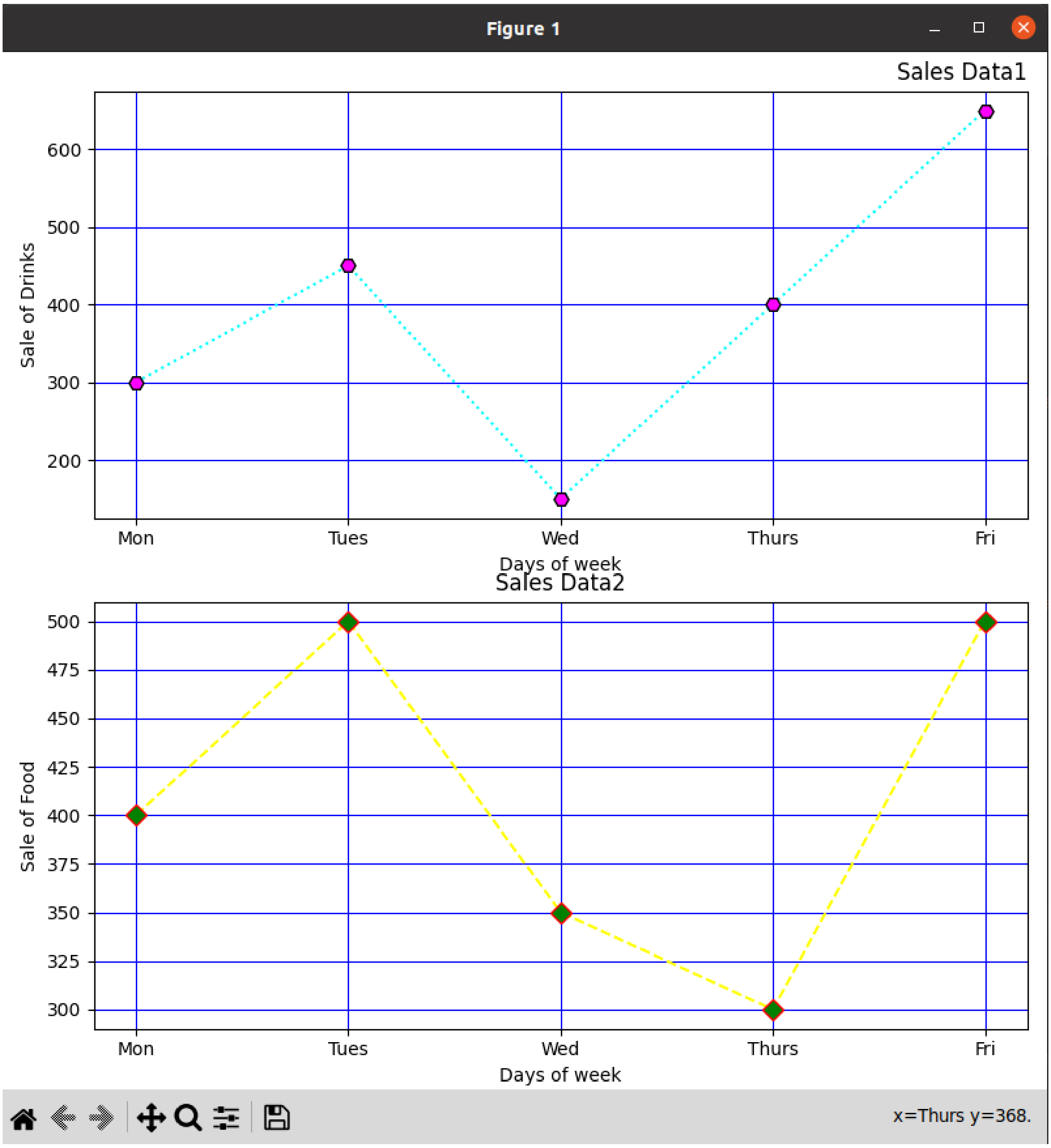


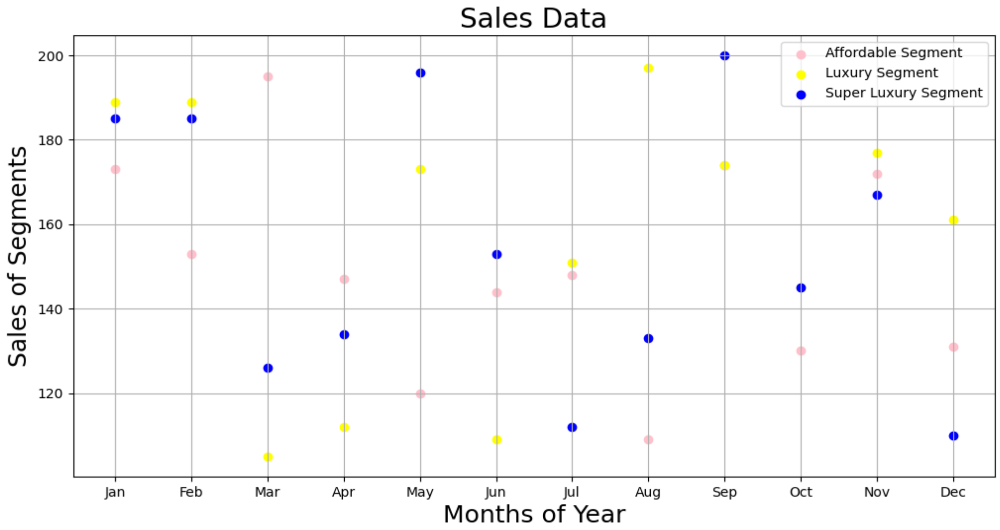
Figure 1
Sales Data1
Sale of Drinks
200
300
400
500
600
Mon
Tues
Wed
Thurs
Fri
Days of week
Sales Data2
Sale of Food
300
325
350
375
400
425
450
475
500
Mon
Tues
Wed
Thurs
Fri
Days of week
x=Thurs y=368.

9. Program to create a scatter plot for the product details.

Input:

```
import matplotlib.pyplot as plt

months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']

affordable_sales = [173, 153, 195, 147, 120, 144, 148, 109, 174, 130, 172, 131]

luxury_sales = [189, 189, 105, 112, 173, 109, 151, 197, 174, 145, 177, 161]

super_luxury_sales = [185, 185, 126, 134, 196, 153, 112, 133, 200, 145, 167, 110]

plt.figure(figsize=(12, 6))

plt.scatter(months, affordable_sales, color='pink', label='Affordable Segment')

plt.scatter(months, luxury_sales, color='yellow', label='Luxury Segment')

plt.scatter(months, super_luxury_sales, color='blue', label='Super Luxury Segment')

plt.xlabel('Months of Year', fontsize=18)

plt.ylabel('Sales of Segments', fontsize=18)

plt.title('Sales Data', fontsize=20)

plt.legend()

plt.grid()

plt.show()
```

Output:

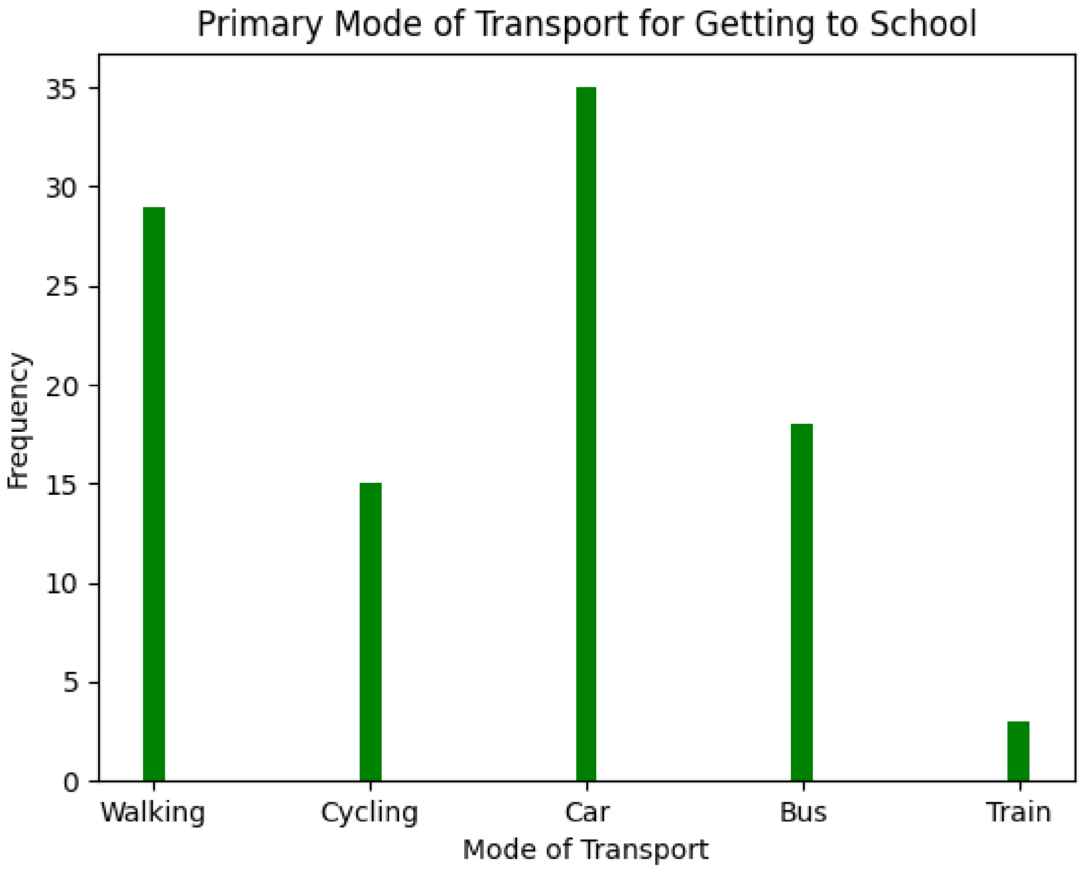


10. Program to create bar chart for given data regarding 'Primary mode of transport'

Input:

```
import matplotlib.pyplot as plt

modes = ["Walking", "Cycling", "Car", "Bus", "Train"]

frequencies = [29, 15, 35, 18, 3]

width = 0.1

color = "green"

plt.bar(modes, frequencies, width=width, color=color)

plt.xlabel("Mode of Transport")

plt.ylabel("Frequency")

plt.title("Primary Mode of Transport for Getting to School")

plt.show()
```

Output:

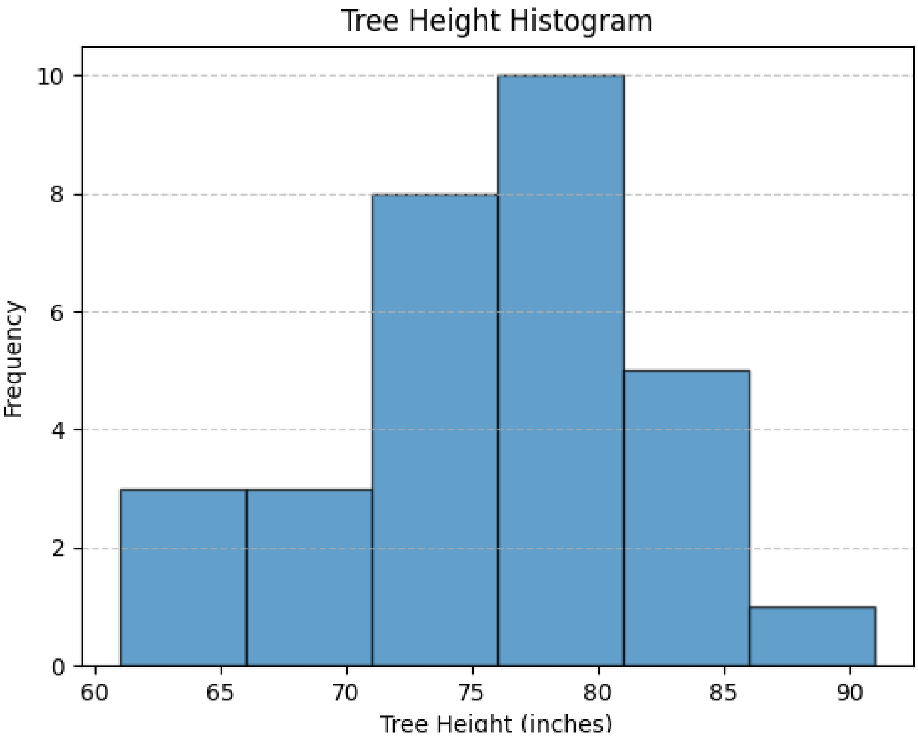


11. Program to create histogram with bin size of 5 for the given data regarding height of cherry trees.

Input:

```
import numpy as np

import matplotlib.pyplot as plt

tree_heights = np.array([61, 63, 64, 66, 68, 69, 71, 71.5, 72, 72.5, 73, 73.5, 74, 74.5, 76, 76.2, 76.5, 77, 77.5, 78, 78.5, 79, 79.2, 80, 81, 82, 83, 84, 85, 87])

bin_size = 5

hist, bins = np.histogram(tree_heights, bins=np.arange(min(tree_heights), max(tree_heights) + bin_size, bin_size))

plt.hist(tree_heights, bins=bins, edgecolor='black', alpha=0.7)

plt.xlabel('Tree Height (inches)')

plt.ylabel('Frequency')

plt.title('Tree Height Histogram')

plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.show()
```

Output:

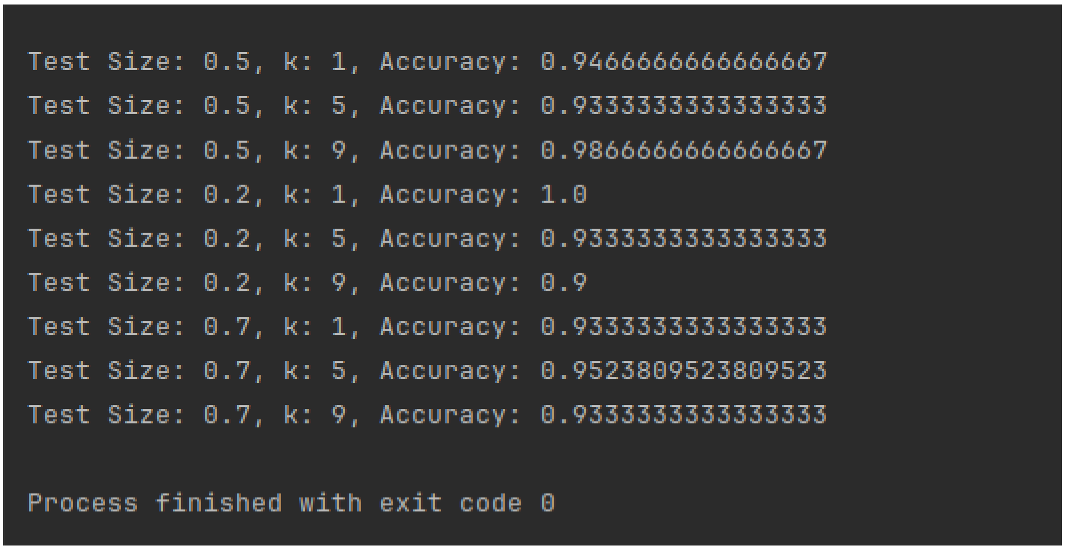


12. Write a program to implement KNN algorithm using iris data Set. Use different values for K and different values for text and training data.

Input:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
print("Name: APARNA MOHAN")
print("Reg No: SJC22MCA-2013")
print("Batch: 22-24")
print()
iris_data = pd.read_csv('iris.csv')

X = iris_data.iloc[:, :-1].values
y = iris_data.iloc[:, -1].values

test_sizes = [0.5, 0.2, 0.7]
k_values = [1,5,9]

for test_size in test_sizes:
    for k in k_values:
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_size)
        knn = KNeighborsClassifier(n_neighbors=k)
        knn.fit(X_train, y_train)
        y_pred = knn.predict(X_test)
        accuracy = accuracy_score(y_test, y_pred)
        print(f'Test Size: {test_size}, k: {k}, Accuracy: {accuracy}')
```

Output:



```
Test Size: 0.5, k: 1, Accuracy: 0.9466666666666667
Test Size: 0.5, k: 5, Accuracy: 0.9333333333333333
Test Size: 0.5, k: 9, Accuracy: 0.9866666666666667
Test Size: 0.2, k: 1, Accuracy: 1.0
Test Size: 0.2, k: 5, Accuracy: 0.9333333333333333
Test Size: 0.2, k: 9, Accuracy: 0.9
Test Size: 0.7, k: 1, Accuracy: 0.9333333333333333
Test Size: 0.7, k: 5, Accuracy: 0.9523809523809523
Test Size: 0.7, k: 9, Accuracy: 0.9333333333333333

Process finished with exit code 0
```

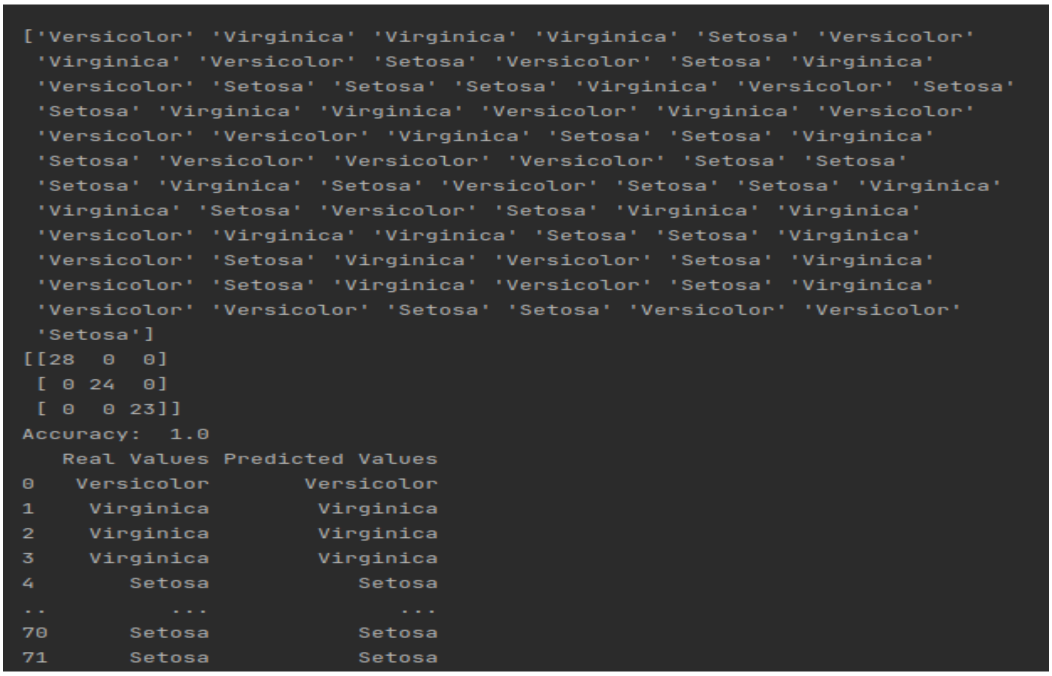
13. Write a program to implement naive bayes classification using different naive Bayes classification algorithms.

Input:

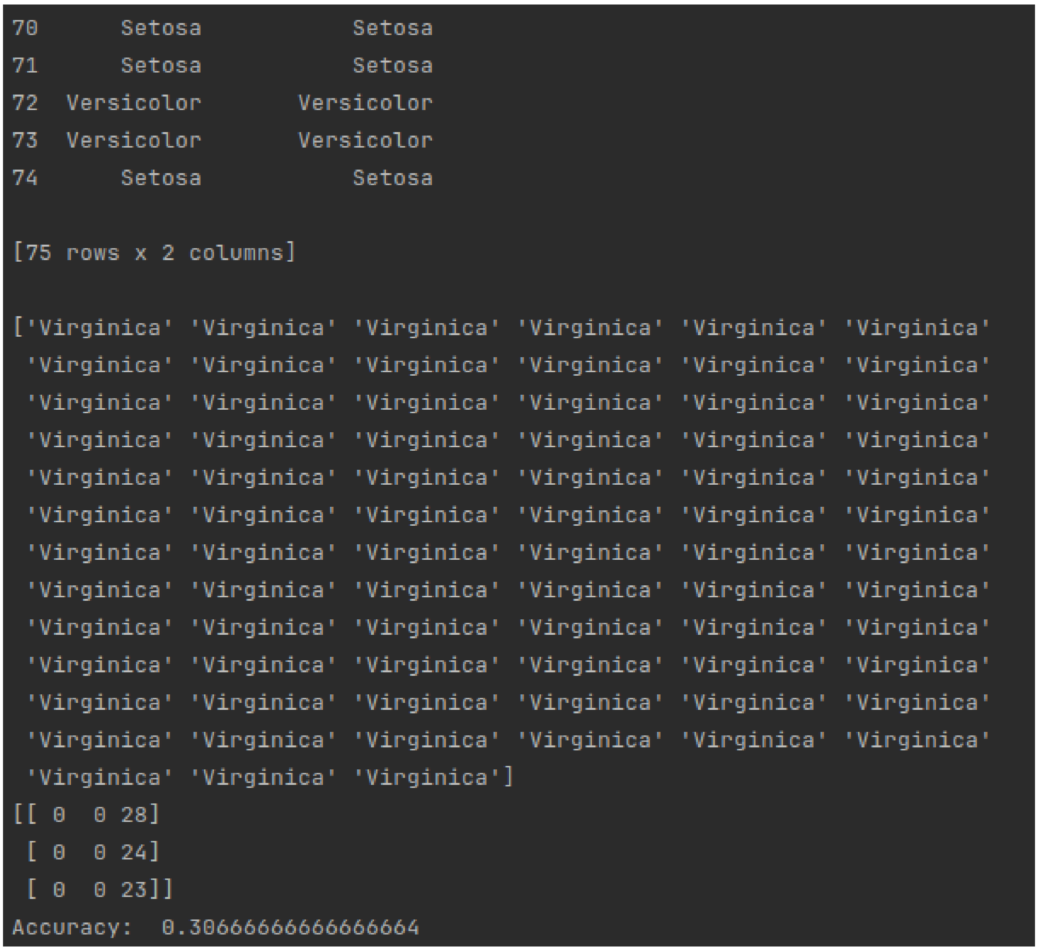
```
import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

print("Name: Aparna Mohan")

print("Reg No: SJC22MCA-2013")

print("Batch: 22-24")

print()

dataset = pd.read_csv('iris.csv')

X = dataset.iloc[:,:4].values

y = dataset['variety'].values

dataset.head (5)

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.5)

from sklearn.naive_bayes import GaussianNB

classifier = GaussianNB ()

classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test)

print(y_pred)

from sklearn.metrics import confusion_matrix

cm =confusion_matrix(y_test, y_pred)

print(cm)

from sklearn.metrics import accuracy_score

print ("Accuracy: ", accuracy_score (y_test, y_pred))

df = pd.DataFrame({'Real Values':y_test, 'Predicted Values':y_pred})
```

```
print(df)

print()

from sklearn.naive_bayes import BernoulliNB

classif = BernoulliNB ()

classif.fit(X_train, y_train)

y_pred = classif.predict(X_test)

print(y_pred)

from sklearn.metrics import confusion_matrix

cmx =confusion_matrix(y_test, y_pred)

print(cmx)

from sklearn.metrics import accuracy_score

print ("Accuracy: ", accuracy_score (y_test, y_pred))

fd = pd.DataFrame({'Real Values':y_test, 'Predicted Values':y_pred})

print(fd)
```

Output:



```
['Versicolor' 'Virginica' 'Virginica' 'Virginica' 'Setosa' 'Versicolor'
 'Virginica' 'Versicolor' 'Setosa' 'Versicolor' 'Setosa' 'Virginica'
 'Versicolor' 'Setosa' 'Setosa' 'Setosa' 'Virginica' 'Versicolor' 'Setosa'
 'Setosa' 'Virginica' 'Virginica' 'Versicolor' 'Virginica' 'Versicolor'
 'Versicolor' 'Versicolor' 'Virginica' 'Setosa' 'Setosa' 'Virginica'
 'Setosa' 'Versicolor' 'Versicolor' 'Versicolor' 'Setosa' 'Setosa'
 'Setosa' 'Virginica' 'Setosa' 'Versicolor' 'Setosa' 'Setosa' 'Virginica'
 'Virginica' 'Setosa' 'Versicolor' 'Setosa' 'Virginica' 'Virginica'
 'Versicolor' 'Virginica' 'Virginica' 'Setosa' 'Setosa' 'Virginica'
 'Versicolor' 'Setosa' 'Virginica' 'Versicolor' 'Setosa' 'Virginica'
 'Versicolor' 'Setosa' 'Virginica' 'Versicolor' 'Setosa' 'Virginica'
 'Versicolor' 'Versicolor' 'Setosa' 'Setosa' 'Versicolor' 'Versicolor'
 'Setosa']
[[28  0  0]
 [ 0 24  0]
 [ 0  0 23]]
Accuracy:  1.0
   Real Values Predicted Values
0   Versicolor       Versicolor
1    Virginica        Virginica
2    Virginica        Virginica
3    Virginica        Virginica
4       Setosa           Setosa
..         ...              ...
70      Setosa           Setosa
71      Setosa           Setosa
```



```
70      Setosa           Setosa
71      Setosa           Setosa
72  Versicolor       Versicolor
73  Versicolor       Versicolor
74      Setosa           Setosa

[75 rows x 2 columns]

['Virginica' 'Virginica' 'Virginica' 'Virginica' 'Virginica' 'Virginica'
 'Virginica' 'Virginica' 'Virginica' 'Virginica' 'Virginica' 'Virginica'
 'Virginica' 'Virginica' 'Virginica' 'Virginica' 'Virginica' 'Virginica'
 'Virginica' 'Virginica' 'Virginica' 'Virginica' 'Virginica' 'Virginica'
 'Virginica' 'Virginica' 'Virginica' 'Virginica' 'Virginica' 'Virginica'
 'Virginica' 'Virginica' 'Virginica' 'Virginica' 'Virginica' 'Virginica'
 'Virginica' 'Virginica' 'Virginica' 'Virginica' 'Virginica' 'Virginica'
 'Virginica' 'Virginica' 'Virginica' 'Virginica' 'Virginica' 'Virginica'
 'Virginica' 'Virginica' 'Virginica' 'Virginica' 'Virginica' 'Virginica'
 'Virginica' 'Virginica' 'Virginica' 'Virginica' 'Virginica' 'Virginica'
 'Virginica' 'Virginica' 'Virginica' 'Virginica' 'Virginica' 'Virginica'
 'Virginica' 'Virginica' 'Virginica' 'Virginica' 'Virginica' 'Virginica'
 'Virginica' 'Virginica' 'Virginica']
[[ 0  0 28]
 [ 0  0 24]
 [ 0  0 23]]
Accuracy:  0.30666666666666664
```

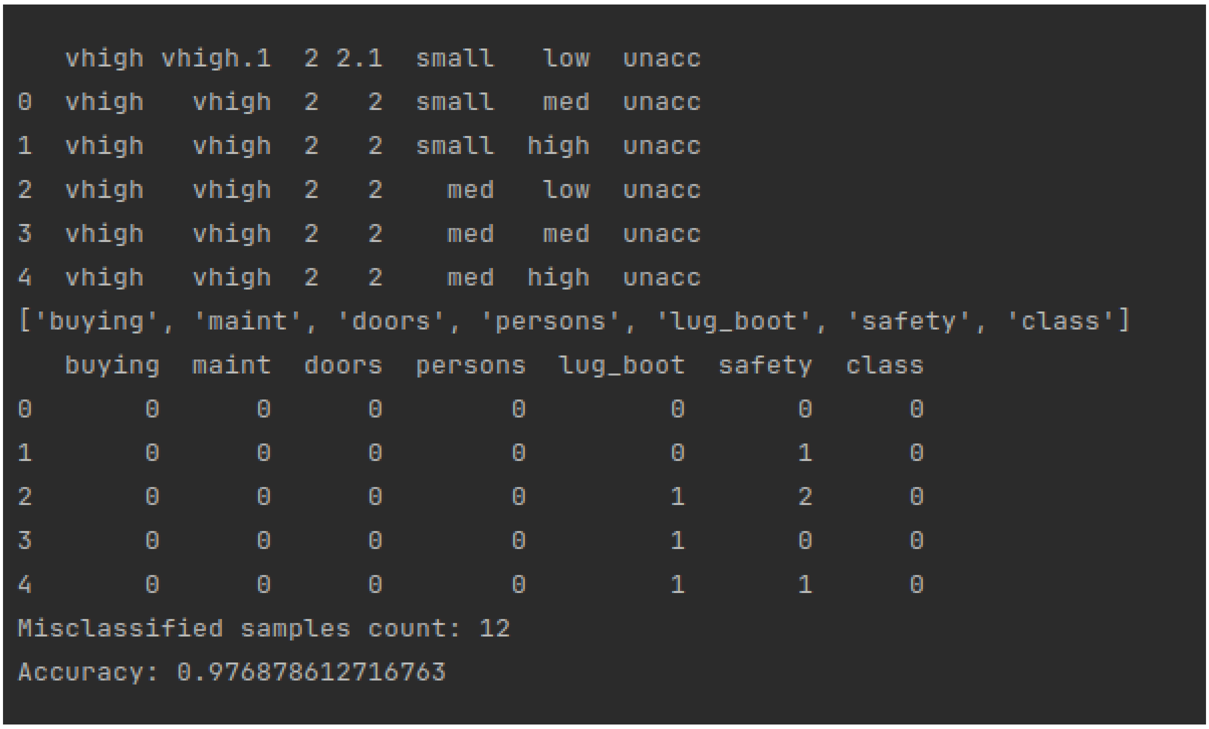
14. Write a program to implement decision tree algorithm using the given data set

Input:

```
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy_score

print("Name: Aparna Mohan")

print("Reg No: SJC22MCA-2013")

print("Batch: 22-24")

print()

data = pd.read_csv('car.csv')

print(data.head())

col_names = ['buying', 'maint', 'doors', 'persons', 'lug_boot', 'safety', 'class']

data.columns = col_names

print(col_names)

data['class'],class_names = pd.factorize(data['class'])

data['buying'],_ = pd.factorize(data['buying'])

data['maint'],_ = pd.factorize(data['maint'])

data['doors'],_ = pd.factorize(data['doors'])

data['persons'],_ = pd.factorize(data['persons'])

data['lug_boot'],_ = pd.factorize(data['lug_boot'])

data['safety'],_ = pd.factorize(data['safety'])

print(data.head())

X = data.iloc[:, :-1]

y = data.iloc[:, -1]
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

tree1 = DecisionTreeClassifier()

tree1.fit(X_train, y_train)

y_pred = tree1.predict(X_test)

# how did our model perform?

count_misclassified = (y_test != y_pred).sum()

print('Misclassified samples count:', count_misclassified)

accuracy = accuracy_score (y_test, y_pred)

print("Accuracy:", accuracy)
```

Output:



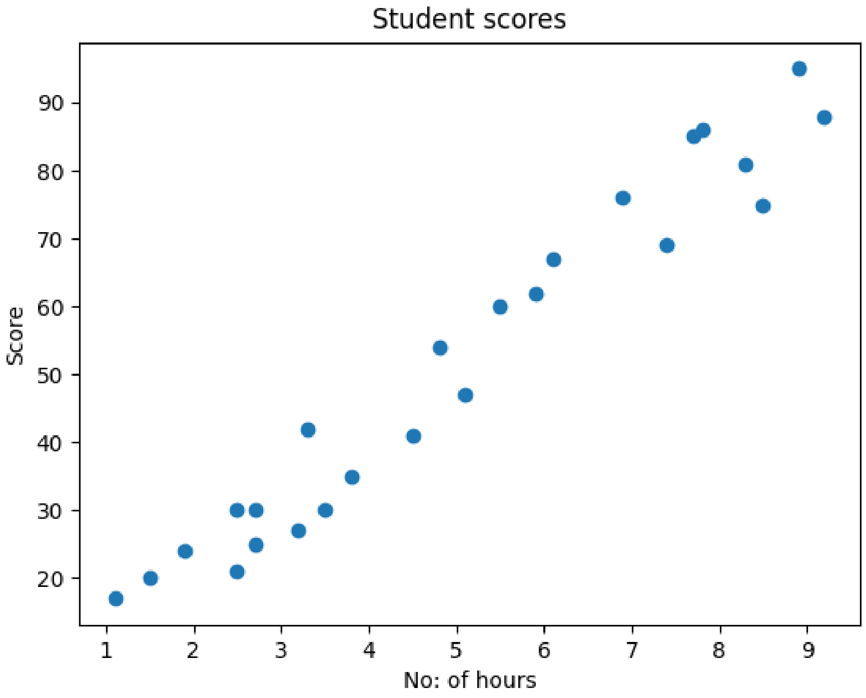
```
   vhigh vhigh.1  2 2.1  small   low  unacc
0  vhigh   vhigh  2   2  small   med  unacc
1  vhigh   vhigh  2   2  small  high  unacc
2  vhigh   vhigh  2   2    med   low  unacc
3  vhigh   vhigh  2   2    med   med  unacc
4  vhigh   vhigh  2   2    med  high  unacc
['buying', 'maint', 'doors', 'persons', 'lug_boot', 'safety', 'class']
   buying  maint  doors  persons  lug_boot  safety  class
0       0      0      0        0         0       0      0
1       0      0      0        0         0       1      0
2       0      0      0        0         1       2      0
3       0      0      0        0         1       0      0
4       0      0      0        0         1       1      0
Misclassified samples count: 12
Accuracy: 0.976878612716763
```

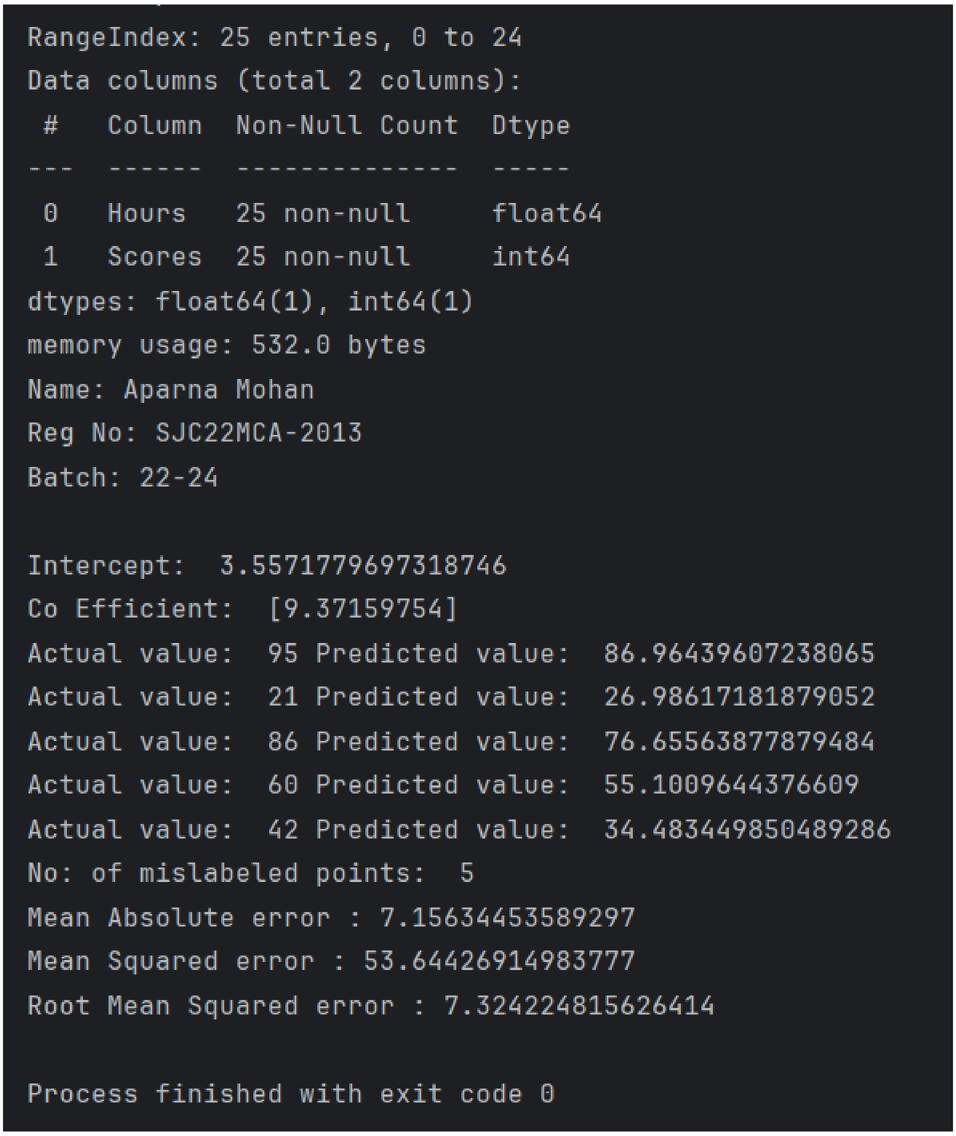
15. Write a program to demonstrate Simple Linear Regression using given data set

Input:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
stud = pd.read_csv('student_scores.csv')
stud.describe()
stud.info()

Xax = stud.iloc[:,0]
Yax = stud.iloc[:,1]
plt.scatter(Xax, Yax)
plt.xlabel("No: of hours")
plt.ylabel("Score")
plt.title("Student scores")
plt.show()
X = stud.iloc[:,:-1]
y = stud.iloc[:, 1]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
#print(X_train)
print("Name: Aparna Mohan")
print("Reg No: SJC22MCA-2013")
print("Batch: 22-24")
print()

reg = LinearRegression()

reg.fit(X_train, y_train)

print('Intercept: ', reg.intercept_)

print('Co Efficient: ', reg.coef_)

y_pred = reg.predict(X_test)

for(i,j) in zip(y_test, y_pred):

    if(i!=j):

        print('Actual value: ', i, 'Predicted value: ',j)

print('No: of mislabeled points: ', (y_test != y_pred).sum())

print("Mean Absolute error :", metrics.mean_absolute_error(y_test,y_pred))

print("Mean Squared error :", metrics.mean_squared_error(y_test,y_pred))

print("Root Mean Squared error :",
np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
```

Output:





```
RangeIndex: 25 entries, 0 to 24
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Hours   25 non-null     float64
 1   Scores  25 non-null     int64  
dtypes: float64(1), int64(1)
memory usage: 532.0 bytes
Name: Aparna Mohan
Reg No: SJC22MCA-2013
Batch: 22-24

Intercept:  3.5571779697318746
Co Efficient:  [9.37159754]
Actual value:  95 Predicted value:  86.96439607238065
Actual value:  21 Predicted value:  26.98617181879052
Actual value:  86 Predicted value:  76.65563877879484
Actual value:  60 Predicted value:  55.1009644376609
Actual value:  42 Predicted value:  34.483449850489286
No: of mislabeled points:  5
Mean Absolute error : 7.15634453589297
Mean Squared error : 53.64426914983777
Root Mean Squared error : 7.324224815626414

Process finished with exit code 0
```

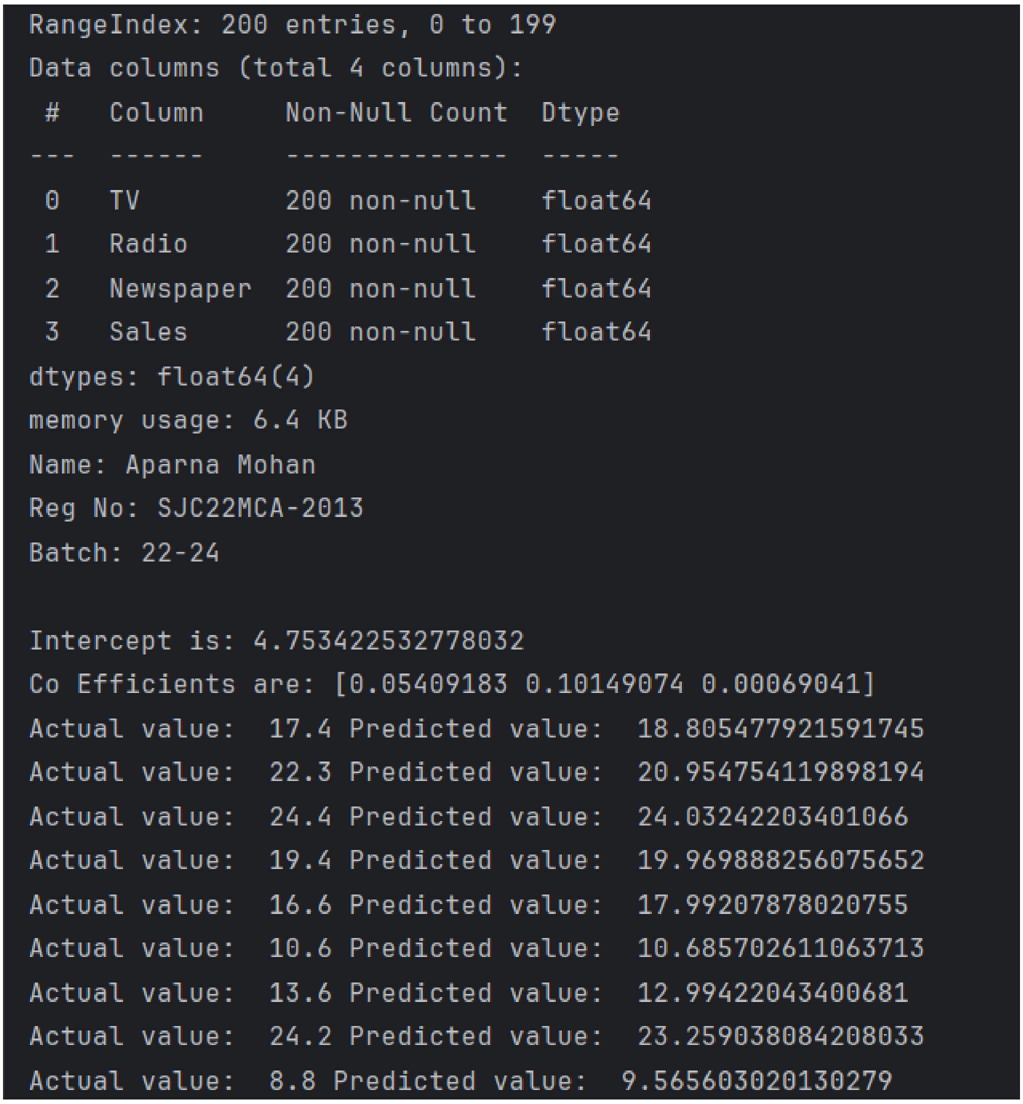
16. Write a program to implement Multiple Linear Regression using appropriate data set

Input:

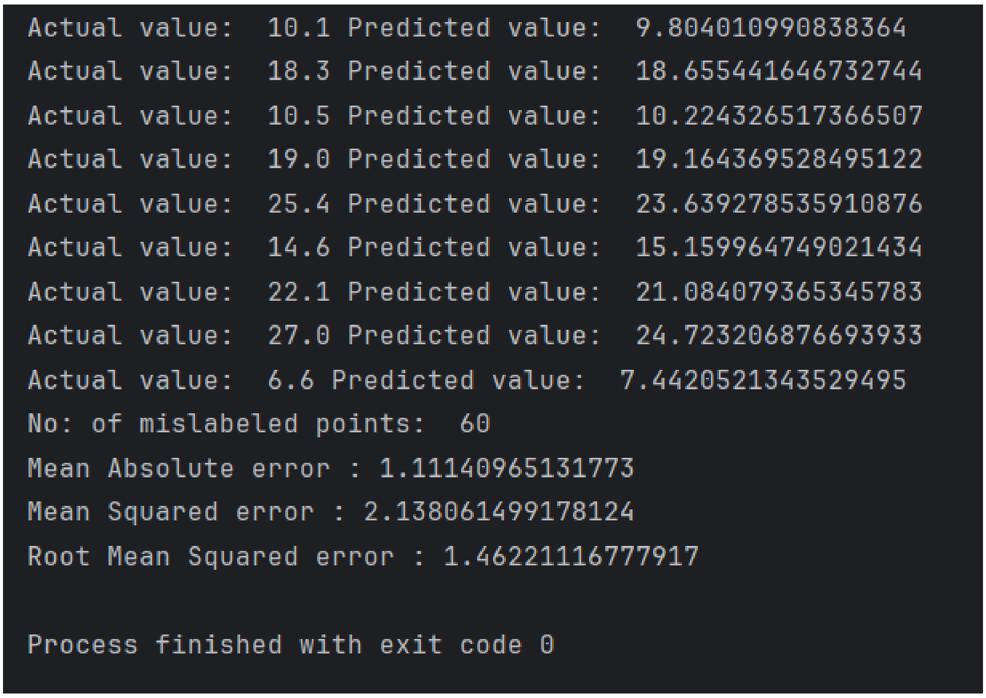
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
advertising = pd.read_csv('Company_data.csv')
advertising.head()
advertising.describe()
advertising.info()
X = advertising.iloc[:, :-1]
y = advertising.iloc[:, -1]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
reg = LinearRegression()
reg.fit(X_train, y_train)
print("Name: Aparna Mohan")
print("Reg No: SJC22MCA-2013")
print("Batch: 22-24")
print()
print('Intercept is:', reg.intercept_)
print('Co Efficients are:', reg.coef_)
y_pred = reg.predict(X_test)
for(i,j) in zip(y_test, y_pred):
    if(i!=j):
```

```
    print('Actual value: ', i, 'Predicted value: ', j)
print('No: of mislabeled points: ', (y_test != y_pred).sum())
print("Mean Absolute error :", metrics.mean_absolute_error(y_test,y_pred))
print("Mean Squared error :", metrics.mean_squared_error(y_test,y_pred))
print("Root Mean Squared error :", np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
```

Output:



```
RangeIndex: 200 entries, 0 to 199
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   TV         200 non-null    float64
 1   Radio      200 non-null    float64
 2   Newspaper  200 non-null    float64
 3   Sales      200 non-null    float64
dtypes: float64(4)
memory usage: 6.4 KB
Name: Aparna Mohan
Reg No: SJC22MCA-2013
Batch: 22-24

Intercept is: 4.753422532778032
Co Efficients are: [0.05409183 0.10149074 0.00069041]
Actual value:  17.4 Predicted value:  18.805477921591745
Actual value:  22.3 Predicted value:  20.954754119898194
Actual value:  24.4 Predicted value:  24.03242203401066
Actual value:  19.4 Predicted value:  19.969888256075652
Actual value:  16.6 Predicted value:  17.99207878020755
Actual value:  10.6 Predicted value:  10.685702611063713
Actual value:  13.6 Predicted value:  12.99422043400681
Actual value:  24.2 Predicted value:  23.259038084208033
Actual value:  8.8 Predicted value:  9.565603020130279
```



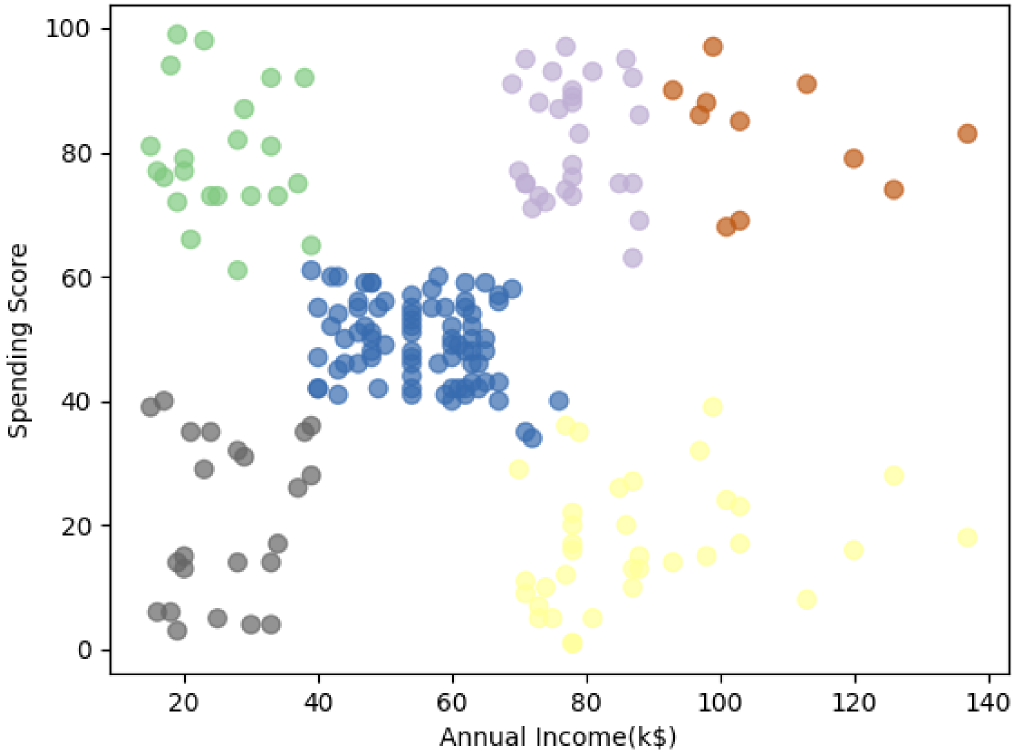
```
Actual value:  10.1 Predicted value:  9.804010990838364
Actual value:  18.3 Predicted value:  18.655441646732744
Actual value:  10.5 Predicted value:  10.224326517366507
Actual value:  19.0 Predicted value:  19.164369528495122
Actual value:  25.4 Predicted value:  23.639278535910876
Actual value:  14.6 Predicted value:  15.159964749021434
Actual value:  22.1 Predicted value:  21.084079365345783
Actual value:  27.0 Predicted value:  24.723206876693933
Actual value:  6.6 Predicted value:  7.4420521343529495
No: of mislabeled points:  60
Mean Absolute error : 1.11140965131773
Mean Squared error : 2.138061499178124
Root Mean Squared error : 1.46221116777917

Process finished with exit code 0
```

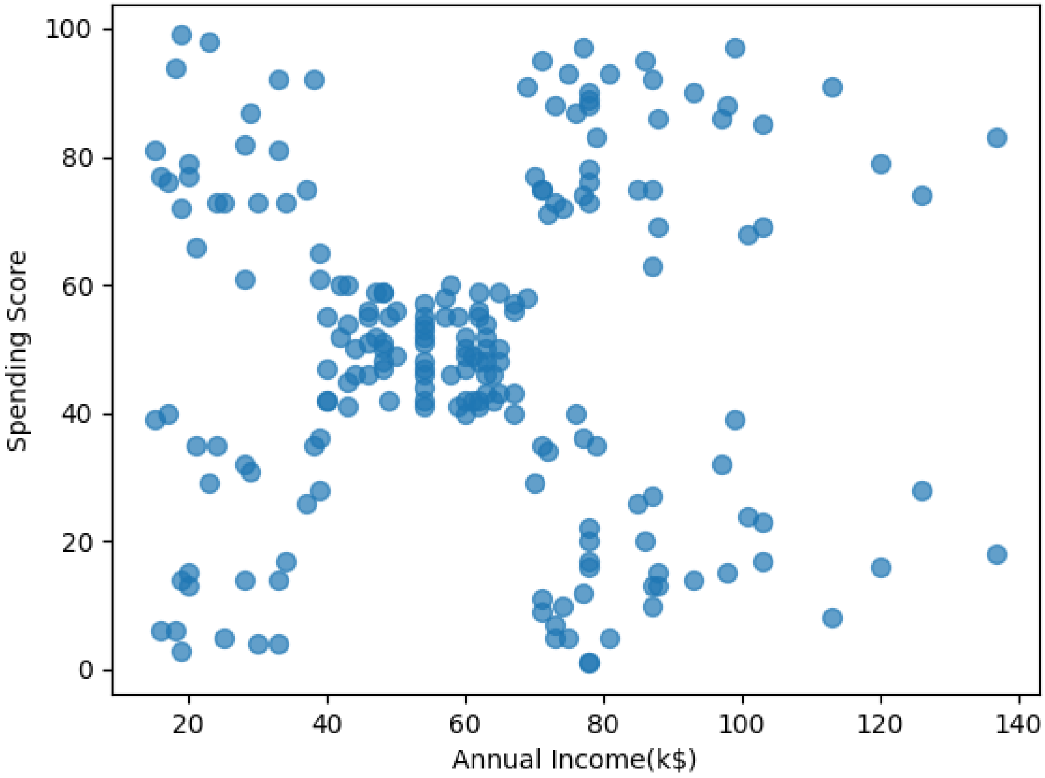
17. Write a program to implement K –Means Clustering Algorithm with k=6. Create a scatter plot to visualize the same.

Input:

```
import pandas as pd

import matplotlib.pyplot as plt

from sklearn.cluster import  KMeans

cust = pd.read_csv('customer_data.csv')

cust.head()

point = cust.iloc[:, 3:5].values

x = point[:, 0]

y = point[:, 1]

plt.scatter(x, y, s=50, alpha=0.7)

plt.xlabel('Annual Income(k$)')

plt.ylabel('Spending Score')

plt.show()

kmeans = KMeans(n_clusters=6, random_state=0)

kmeans.fit(point)

pred_clust_index =kmeans.predict(point)

plt.scatter(x, y, c=pred_clust_index, s=50, alpha=0.7, cmap='virdis')

plt.xlabel('Annual Income(k$)')

plt.ylabel('Spending Score')

plt.show()

center = kmeans.cluster_centers_

plt.scatter(center[:, 0], center[:, 1], c='red', s=100)

plt.xlabel('Annual Income(k$)')

plt.ylabel('Spending Score')

plt.show()
```

Output:


Spending Score
Annual Income(k$)
0
20
40
60
80
100
20
40
60
80
100
120
140

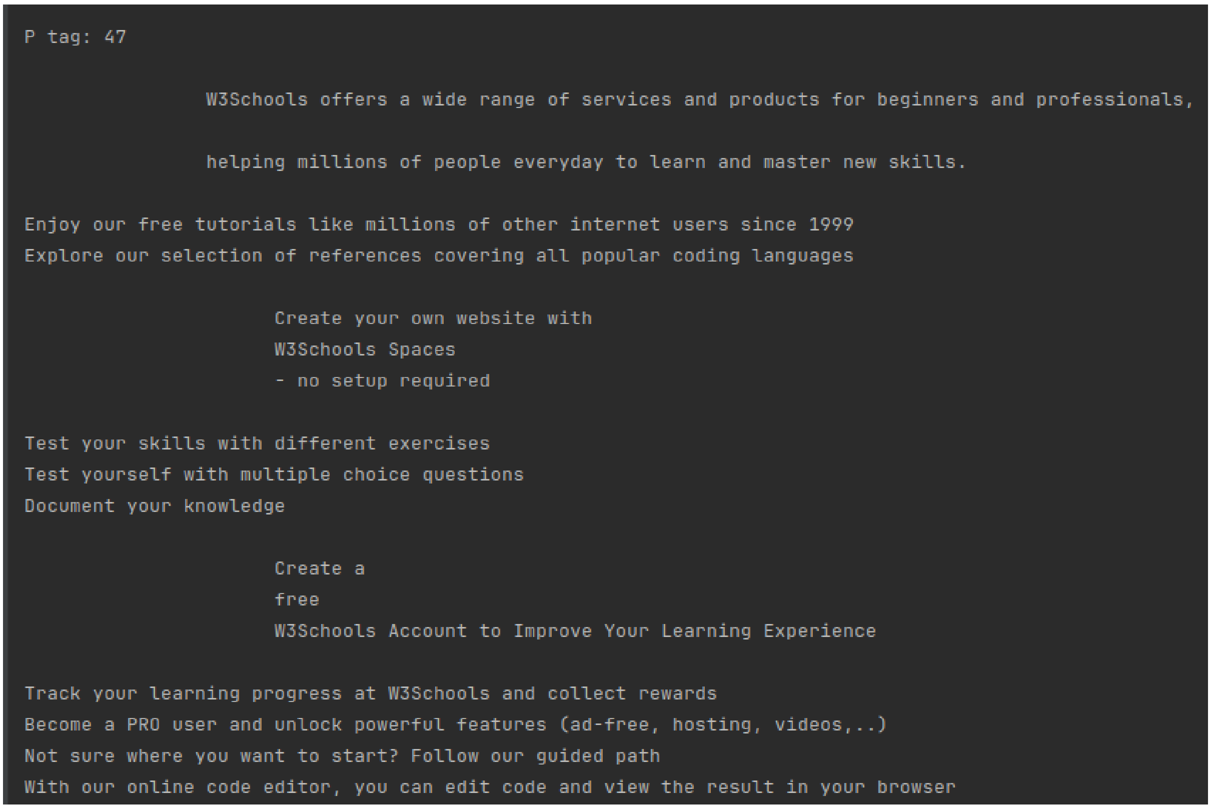

Spending Score
Annual Income(k$)
0
20
40
60
80
100
20
40
60
80
100
120
140

18. Write a program to implement simple web crawler using Python. Extract and display the content of the page(p tag)

Input:

```
import requests

from bs4 import BeautifulSoup

def getdata(url):

    r = requests.get(url)

    return r.content

htmldata = getdata("https://www.w3schools.com/python/python_ml_scale.asp")

soup = BeautifulSoup(htmldata, 'html.parser')

data = ''

print("Name: Aparna Mohan")

print("Reg No: SJC22MCA-2013")

print("Batch: 22-24")

print()

pr = len(soup.find_all('p'))

print("P tag:", pr)

for data in soup.find_all('p'):

    print(data.get_text())
```

Output:



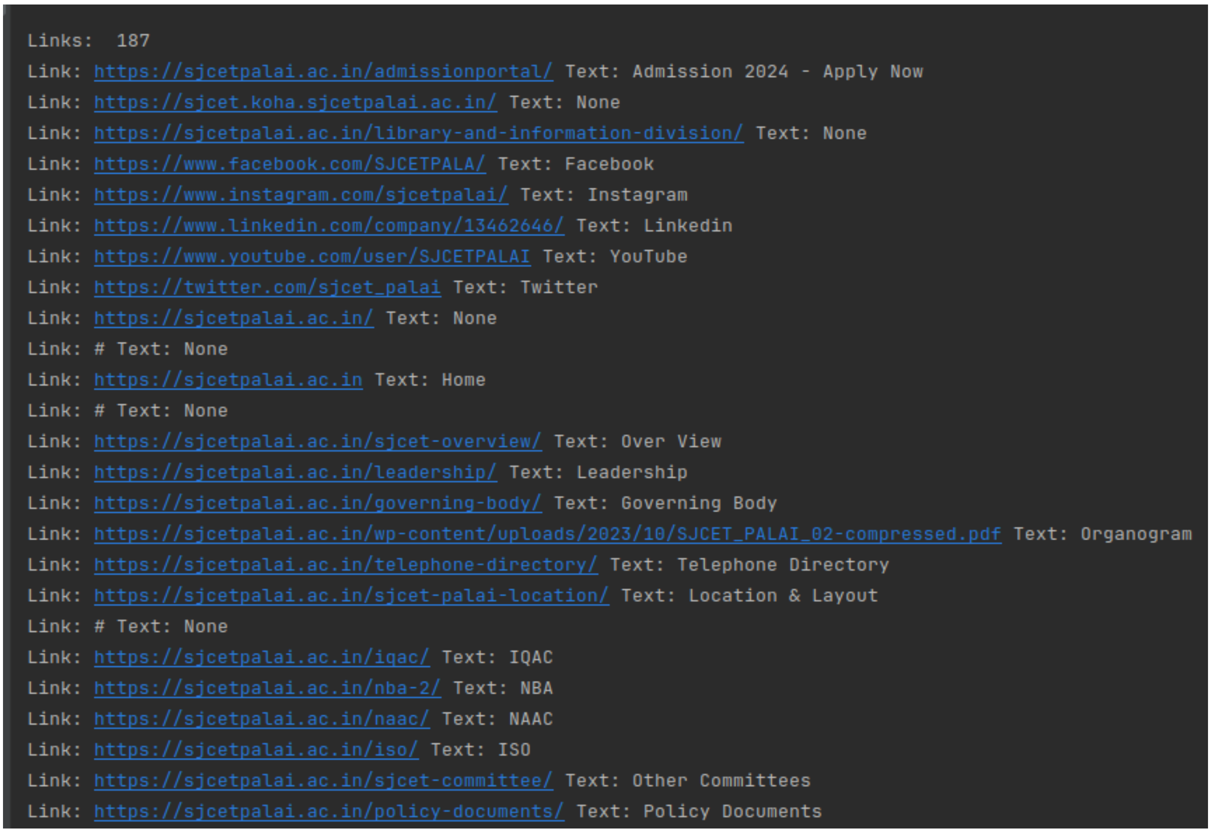
```
P tag: 47

                W3Schools offers a wide range of services and products for beginners and professionals,

                helping millions of people everyday to learn and master new skills.

Enjoy our free tutorials like millions of other internet users since 1999
Explore our selection of references covering all popular coding languages

                      Create your own website with
                      W3Schools Spaces
                      - no setup required

Test your skills with different exercises
Test yourself with multiple choice questions
Document your knowledge

                      Create a
                      free
                      W3Schools Account to Improve Your Learning Experience

Track your learning progress at W3Schools and collect rewards
Become a PRO user and unlock powerful features (ad-free, hosting, videos,..)
Not sure where you want to start? Follow our guided path
With our online code editor, you can edit code and view the result in your browser
```

19. Write a program to implement simple web crawler using Python. Display all hyperlinks in the page

Input:

```
import requests

from bs4 import BeautifulSoup

def getdata(url):

    r = requests.get(url)

    return r.content

htmldata = getdata("https://sjcetpalai.ac.in/")

soup = BeautifulSoup(htmldata, 'html.parser')

print("Name: Aparna Mohan")

print("Reg No: SJC22MCA-2013")

print("Batch: 22-24")

print()

links = soup.find_all("a")

print("Links: ", len(links))

for link in links:

    if link.get("href") != "":

        print("Link:", link.get("href"), "Text:", link.string)
```

Output:



```
Links:  187
Link: https://sjcetpalai.ac.in/admissionportal/ Text: Admission 2024 - Apply Now
Link: https://sjcet.koha.sjcetpalai.ac.in/ Text: None
Link: https://sjcetpalai.ac.in/library-and-information-division/ Text: None
Link: https://www.facebook.com/SJCETPALA/ Text: Facebook
Link: https://www.instagram.com/sjcetpalai/ Text: Instagram
Link: https://www.linkedin.com/company/13462646/ Text: Linkedin
Link: https://www.youtube.com/user/SJCETPALAI Text: YouTube
Link: https://twitter.com/sjcet_palai Text: Twitter
Link: https://sjcetpalai.ac.in/ Text: None
Link: # Text: None
Link: https://sjcetpalai.ac.in Text: Home
Link: # Text: None
Link: https://sjcetpalai.ac.in/sjcet-overview/ Text: Over View
Link: https://sjcetpalai.ac.in/leadership/ Text: Leadership
Link: https://sjcetpalai.ac.in/governing-body/ Text: Governing Body
Link: https://sjcetpalai.ac.in/wp-content/uploads/2023/10/SJCET_PALAI_02-compressed.pdf Text: Organogram
Link: https://sjcetpalai.ac.in/telephone-directory/ Text: Telephone Directory
Link: https://sjcetpalai.ac.in/sjcet-palai-location/ Text: Location & Layout
Link: # Text: None
Link: https://sjcetpalai.ac.in/iqac/ Text: IQAC
Link: https://sjcetpalai.ac.in/nba-2/ Text: NBA
Link: https://sjcetpalai.ac.in/naac/ Text: NAAC
Link: https://sjcetpalai.ac.in/iso/ Text: ISO
Link: https://sjcetpalai.ac.in/sjcet-committee/ Text: Other Committees
Link: https://sjcetpalai.ac.in/policy-documents/ Text: Policy Documents
```

20. Program for Natural Language Processing which performs n-grams

Input:

Using nltk library:

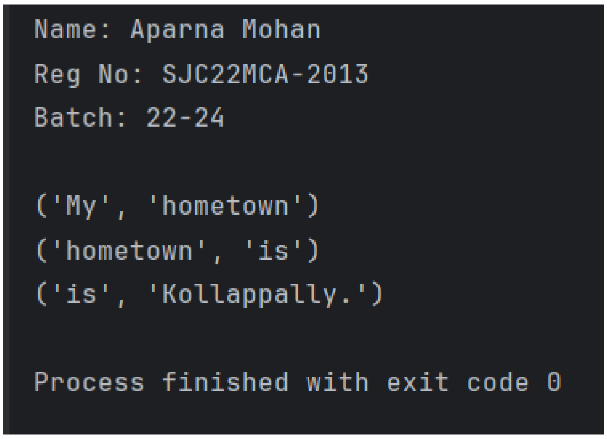
```
print("Name: Aparna Mohan")

print("Reg No: SJC22MCA-2013")

print("Batch: 22-24")

print()

from nltk import ngrams

sent = "My hometown is Kollappally."

n = 2

unigrams = ngrams(sent.split(), n)

for grams in unigrams:

    print(grams)
```

Without using library:

```
def gen_ngrams(text, WordsToCombine):

    words = text.split()

    output = []

    for i in range(len(words) - WordsToCombine + 1):

        output.append(words[i:i + WordsToCombine])

    return output

print("Name: Aparna Mohan")

print("Reg No: SJC22MCA-2013")

print("Batch: 22-24")

print()

x = gen_ngrams(
    text= 'The data set given satisfies the requirement for model generation and s
used in Data Science Lab',
    WordsToCombine=3)
print(x)
```

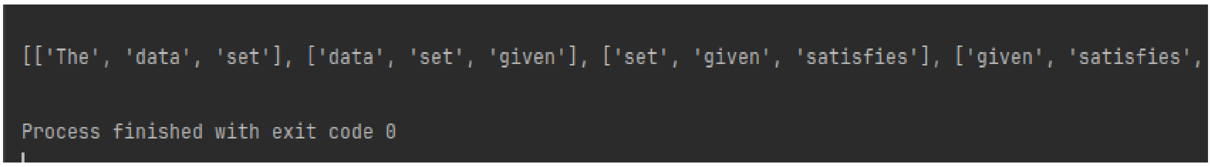
Output:

Using nltk library:



```
Name: Aparna Mohan
Reg No: SJC22MCA-2013
Batch: 22-24

('My', 'hometown')
('hometown', 'is')
('is', 'Kollappally.')

Process finished with exit code 0
```

Without using library:



```
[['The', 'data', 'set'], ['data', 'set', 'given'], ['set', 'given', 'satisfies'], ['given', 'satisfies',

Process finished with exit code 0
```

21. For given text,

I. perform word and sentence tokenization

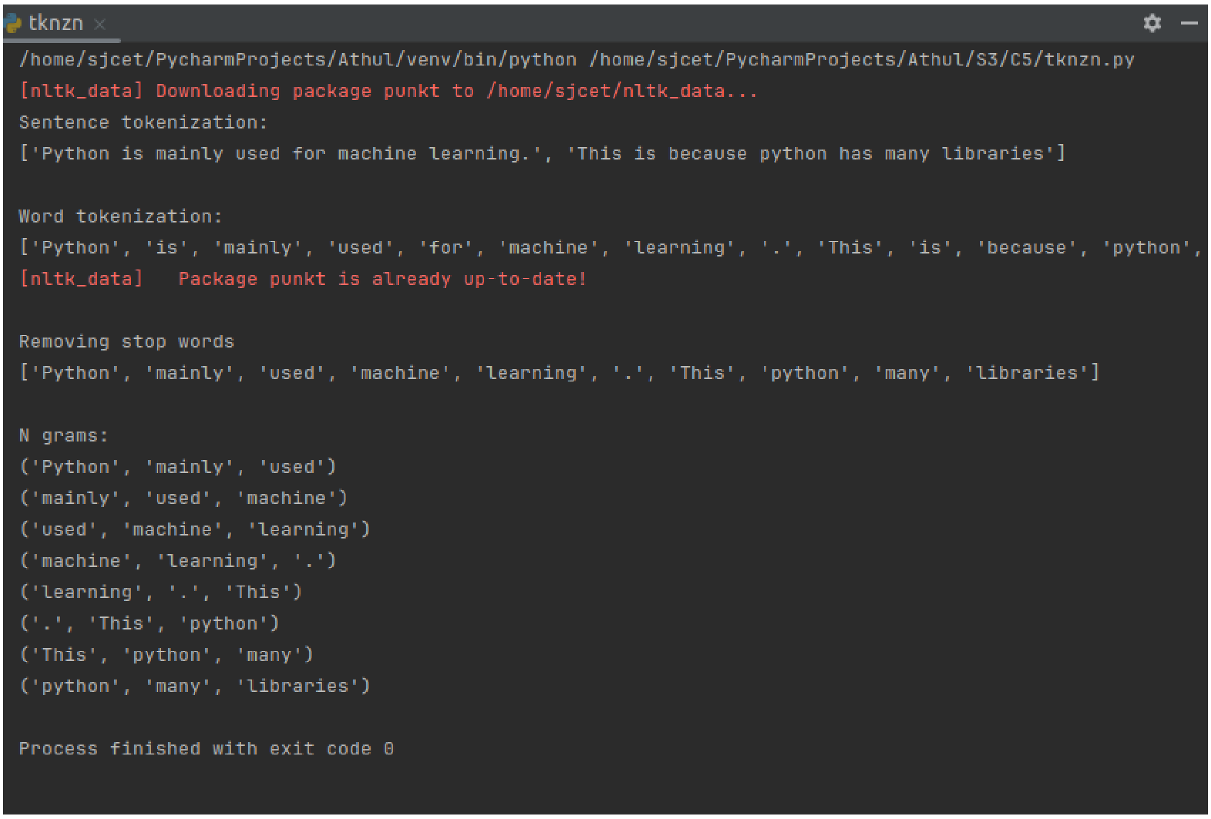
II. Remove the stop words from the given text

III. create n-grams

Input:

```
import nltk

from nltk import ngrams

from nltk.corpus import stopwords

from nltk.tokenize import sent_tokenize, word_tokenize

nltk.download('punkt')

txt1 = 'Python is mainly used for machine learning. This is because python has many libraries'

print('Sentence tokenization: ')

print(sent_tokenize(txt1))

print()

print('Word tokenization: ')

print(word_tokenize(txt1))

text = word_tokenize(txt1)

txt2 = [word for word in text if word not in stopwords.words('english')]

print()

print('Removing stop words')

print(txt2)

print()

print('N grams: ')

unigrams = ngrams(txt2, 3)

for grams in unigrams:

    print(grams)
```

Output:



```
/home/sjcet/PycharmProjects/Athul/venv/bin/python /home/sjcet/PycharmProjects/Athul/S3/C5/tknzn.py
[nltk_data] Downloading package punkt to /home/sjcet/nltk_data...
Sentence tokenization:
['Python is mainly used for machine learning.', 'This is because python has many libraries']

Word tokenization:
['Python', 'is', 'mainly', 'used', 'for', 'machine', 'learning', '.', 'This', 'is', 'because', 'python',
[nltk_data]   Package punkt is already up-to-date!

Removing stop words
['Python', 'mainly', 'used', 'machine', 'learning', '.', 'This', 'python', 'many', 'libraries']

N grams:
('Python', 'mainly', 'used')
('mainly', 'used', 'machine')
('used', 'machine', 'learning')
('machine', 'learning', '.')
('learning', '.', 'This')
('.', 'This', 'python')
('This', 'python', 'many')
('python', 'many', 'libraries')

Process finished with exit code 0
```