CONTENTS

Program: 1

Aim:

Review of python programming – Programs review the fundamentals of python

1. Write a python program that takes an input ‘n’ and calculate n+nn+nnn

   Program Code:

```
n=int(input("Enter the number : "))
print(n+n*n+n*n*n)
```

2. Write a python program to find the largest number from a given list?

   Program Code:

```
l1=[5,7,2,1]
print("list is ",l1)
print("Largest number is ",max(l1))
```

3. Write a python program to copy a given list?

   Program Code:

```
l1=[5,7,2,1]
print("list is ",l1)
l2=l1.copy()
print("copied list is :",l2)
```

4. Write a python program to shuffle a given list?

   Program Code:

```
l3=[1,2,3,4]
print("list",l3)
random.shuffle(l3)
print("Shuffled list",l3)
```

5. Write a python program to sort a given dictionary by value?

   Program Code:

```
dict1 = {1: 1, 2: 9, 3: 4}
print("Dictionary is ",dict1)
sorted_values = sorted(dict1.values())
```

```
sorted_dict = {}
for i in sorted_values:
    for k in dict1.keys():
        if dict1[k] == i:
            sorted_dict[k] = dict1[k]
            break
print("Sorted dictionary is ",sorted_dict)
```

6. Write a python program to add values to a dictionary?

Program Code:

```
dic={1:"annu",2:"binu"}
print("Dictionary is ",dic)
dic[3]="annu"
dic[5]="binu"
print("updated dictionary",dic)
```

7. Write a python program to merge two dictionary?

Program Code:

```
d1={'a':10,'b':20,'c':30}
d2={'z':50,'x':45,'y':78}
print("directory 1",d1)
print("directory 2",d2)
d2.update(d1)
print("Merged directory",d2)
```

Output

1.
```
Enter the number : 4
84
```

2.
```
list is  [5, 7, 2, 1]
Largest number is  7
```

3.
```
list is  [5, 7, 3, 1]
copied list is : [5, 7, 3, 1]
```

4.
```
list is  [1, 2, 3, 4]
Shuffled list [4, 1, 2, 3]
```

5.
```
Dictionary is  {1: 1, 2: 9, 3: 4}
Sorted dictionary is  {1: 1, 3: 4, 2: 9}
```

6.
```
Dictionary is  {1: 'annu', 2: 'binu'}
updated dictionary {1: 'annu', 2: 'binu', 3: 'annu', 5: 'binu'}
```

7.
```
directory 1 {'a': 10, 'b': 20, 'c': 30}
directory 2 {'z': 50, 'x': 45, 'y': 78}
Merged directory {'z': 50, 'x': 45, 'y': 78, 'a': 10, 'b': 20, 'c': 30}
```

Program: 2

Aim:

Perform all matrix operation using python (using numpy)

Program Code:

```
import numpy as np
mat1=np.array([[12,23,22],[5,87,34],[44,77,3]])
mat2=np.array([[12,32,22],[5,78,43],[44,77,3]])
print('ADDITION')
print(np.add(mat1,mat2))
print('SUBTRACTION')
print(np.subtract(mat1,mat2))
print('DIVISION')
print(np.divide(mat1,mat2))
print('MULTIPLICATION')
print(np.multiply(mat1,mat2))
print('DOT PRODUCT')
print(np.dot(mat1,mat2))
print('SQUARE ROOT')
print(np.sqrt(mat1))
print('SUMMATION')
print(np.sum(mat1))
print('TRANSPOSE using "T" ')
print(mat1.T)
print('TRANSPOSE using transpose function ')
print(np.transpose(mat1))
```

Output

```
C:\Users\mca\PycharmProjectspython\pythonProject2\venv\Scripts
ADDITION
[[ 24  55  44]
 [ 10 165  77]
 [ 88 154   6]]
SUBTRACTION
[[ 0 -9  0]
 [ 0  9 -9]
 [ 0  0  0]]
DIVISION
[[1.         0.71875    1.        ]
 [1.         1.11538462 0.79069767]
 [1.         1.         1.        ]]
MULTIPLICATION
[[ 144  736  484]
 [  25 6786 1462]
 [1936 5929    9]]
DOT PRODUCT
[[1227 3872 1319]
 [1991 9564 3953]
 [1045 7645 4288]]
```

```
SQUARE ROOT
[[3.46410162 4.79583152 4.69041576]
 [2.23606798 9.32737905 5.83095189]
 [6.63324958 8.77496439 1.73205081]]
SUMMATION
307
TRANSPOSE using "T"
[[12  5 44]
 [23 87 77]
 [22 34  3]]
TRANSPOSE using transpose function
[[12  5 44]
 [23 87 77]
 [22 34  3]]

Process finished with exit code 0
```

Program: 3

Aim:

Program to Perform SVD (Singular Value Decomposition) in Python

Program Code:

```
# Singular-value decomposition
from numpy import array
from scipy.linalg import svd
# define a matrix
A = array([[3, 1, 1], [-1, 3, 1]])
print("A=",A)
# SVD
U, S, V_T = svd(A)
# left singular vectors
print("U=")
print(U)
# singular values
print("S=")
print(S)
#right singular vectors
print("V_T=")
print(V_T)
```

Output

```
C:\Users\mca\PycharmProjectspython\pythonProject2\venv\Scripts\p
A= [[ 3  1  1]
 [-1  3  1]]
U=
[[-0.70710678 -0.70710678]
 [-0.70710678  0.70710678]]
S=
[3.46410162 3.16227766]
V_T=
[[-4.08248290e-01 -8.16496581e-01 -4.08248290e-01]
 [-8.94427191e-01  4.47213595e-01  5.26260748e-16]
 [-1.82574186e-01 -3.65148372e-01  9.12870929e-01]]

Process finished with exit code 0
```

.

Program: 4

Aim:

Programs using matplotlib / plotly / bokeh / seaborn for data visualisation.

Program Code:

1. Class stratified Histogram(using matplotlib)

```
import numpy as np
import matplotlib.pyplot as plt
bins = 10
data = np.random.randn(1000, 3)
colors = ['blue','green', 'red']
plt.hist(data, bins, histtype='bar', color=colors, stacked=True, fill=True)
plt.show()
```

2. Multiple Scatter matrix(using plotly)

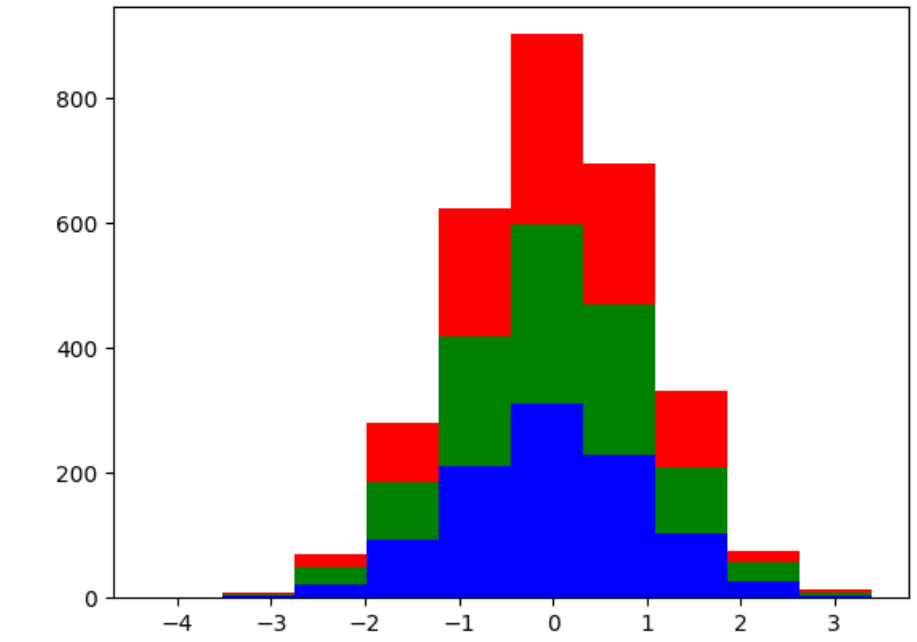
```
import plotly.express as px df = px.data.iris()
fig = px.scatter_matrix(df,
dimensions=["sepal_width", "sepal_length", "petal_width", "petal_length"],
color="species", symbol="species", title="Scatter matrix of iris data set",
labels={col:col.replace('_', ' ') for col in df.columns}) # remove underscore
fig.update_traces(diagonal_visible=False) fig.show()
```

3. Histogram(using seaborn)

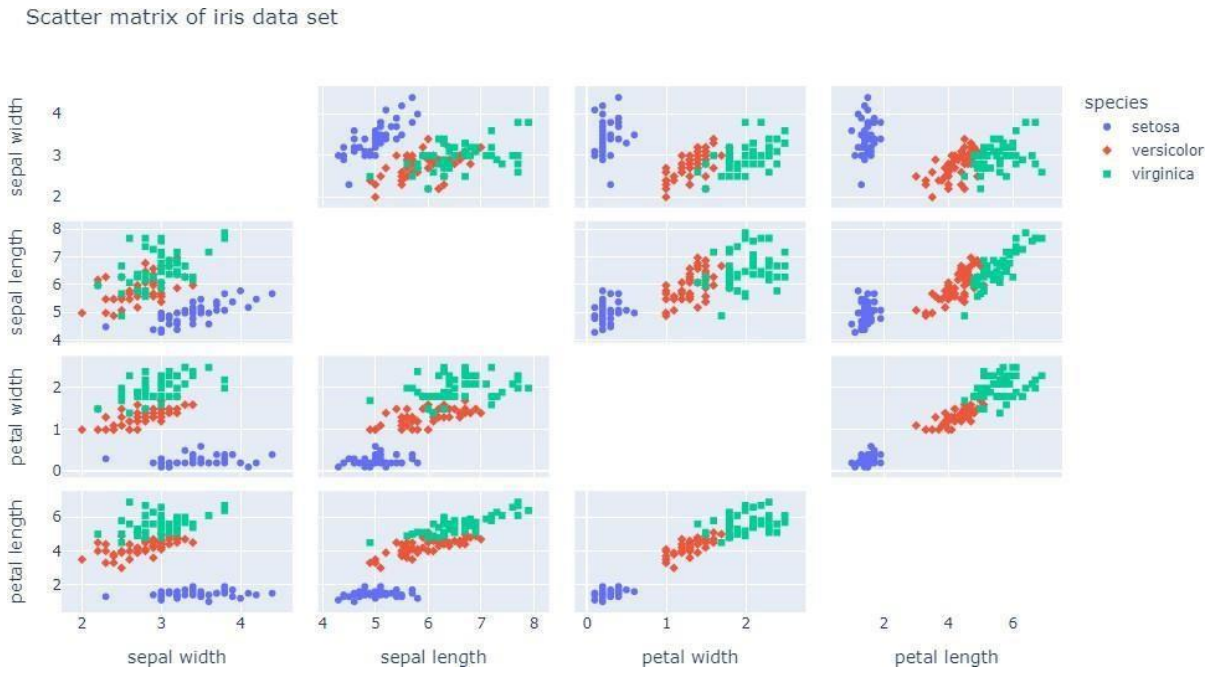
```
import matplotlib.pyplot as plt import pandas as pd
import numpy as np import seaborn as sns
# Load the data
df = pd.read_csv('iris.csv')
# Extract feature we're interested in data = df['SepalWidthCm']
# Generate histogram/distribution plot sns.displot(data)
plt.show()
```

Output

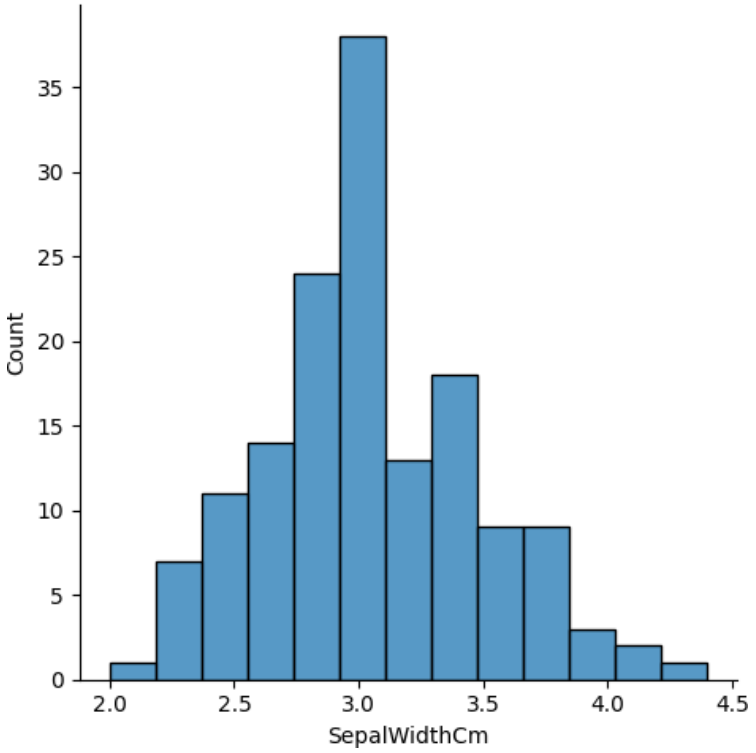
1. Class stratified Histogram



2. Multiple Scatter matrix



3. Histogram



Program: 5

Aim:

Programs to handle data using pandas.

Program Code:

```
#python program to demonstrate
#Working of array using pandas
import pandas as pd

#declare first array
student_dict={'Name':['Joe','Nat','vimal'],'Age':[20,21,20]}
student_df=pd.DataFrame(student_dict)
print(student_df)

#declare second array
marks_dict={'Marks':[85.10,77.80,83.23]}
mark_df=pd.DataFrame(marks_dict)
print(mark_df)

#join two array

joined_df=student_df.join(mark_df)
print(joined_df)

student_dict={'Name':['Joe','Nat','Harrry'],'Age':[20,21,19],'Marks':[85.10,77.80,91.54]}
student_df=pd.DataFrame(student_dict)
print(student_df)

#select top 2 rows
print(student_df.head(1))
#select last 2 rows
print(student_df.tail(1))

#select value at row index 0 and column'Name'
print(student_df.at[0,'Name'])

#select value at first row and first column
print(student_df.iat[1,0])

#select values of 'Name' column
print(student_df.get('Name'))
```

```
#select values from row index 0 to 2 and 'Name' column
print(student_df.loc[0:2,['Name']])

#sort column by marks
student_df=student_df.sort_values(by=['Marks'])
print(student_df)

#select values fro, row index 0 to 2 (exclusive) and column position 0 to 2 exclusive
print(student_df.iloc[0:2,0:2])

#convert dataframe to dict
dict=student_df.to_dict()
print(dict)

#filter a data based on some condition with mark>80
filter=student_df['Marks']>80
student_df['Marks'].where(filter,other=0,inplace=True)
print(student_df)

#filter in names first name start with  'N' then remaining

student_nu=student_df.filter(like='N',axis='columns')
print(student_nu)
```

Output

```
C:\Users\Student\PycharmProjects\
    Name  Age
0    Joe   20
1    Nat   21
2  vimal   20
   Marks
0  85.10
1  77.80
2  83.23
    Name  Age  Marks
0    Joe   20  85.10
1    Nat   21  77.80
2  vimal   20  83.23
```

```
     Name  Age  Marks
0     Joe   20  85.10
1     Nat   21  77.80
2  Harrry   19  91.54
  Name  Age  Marks
0  Joe   20   85.1
     Name  Age  Marks
2  Harrry   19  91.54
Joe
Nat
0       Joe
1       Nat
2    Harrry
Name: Name, dtype: object
     Name
0     Joe
1     Nat
2  Harrry
     Name  Age  Marks
1     Nat   21  77.80
0     Joe   20  85.10
2  Harrry   19  91.54
```

```
  Name  Age
1  Nat   21
0  Joe   20
{'Name': {1: 'Nat', 0: 'Joe', 2: 'Harrry'}, 'Age': {1: 21, 0: 20, 2: 19}, 'Marks': {1: 77.8, 0: 85.1, 2: 91.54}}
     Name  Age  Marks
1     Nat   21   0.00
0     Joe   20  85.10
2  Harrry   19  91.54
     Name
1     Nat
0     Joe
2  Harrry

Process finished with exit code 0
```

Program: 6

Aim:

Program to implement k-NN classification using any standard dataset available in the public domain and find the accuracy of the algorithm.

Program Code:

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris
from sklearn.metrics import accuracy_score
irisData = load_iris()
m = irisData.data
n = irisData.target
m_train, m_test, n_train, n_test = train_test_split(m,n, test_size=0.2, random_state=46)
knn = KNeighborsClassifier(n_neighbors=2)
knn.fit(m_train, n_train)
print(knn.predict(m_test))
p = knn.predict(m_test)
q = accuracy_score(n_test, p)
print("accuracy of the algorithm is:", q)
```

Output

```
C:\Users\Student\PycharmProjects\pythonProject\venv\Scripts\python.exe
[0 1 1 0 0 2 0 1 1 1 1 2 0 2 0 0 0 0 1 2 0 2 0 1 2 0 1 1 2 2]
accuracy of the algorithm is: 0.9

Process finished with exit code 0
```

Program: 7

Aim:

Program to implement kNN classification using any random data, without using inbuilt package.

Program Code:

```
from math import sqrt
# calculate the Euclidean distance between two vectors    distance = 0.0
def euclidean_distance(row1, row2):
    distance = 0.0
    for i in range(len(row1) - 1):
        distance += (row1[i] - row2[i]) ** 2
    return sqrt(distance)
# Locate the closest neighbors
def get_neighbors(train, test_row, num_neighbors):
    distances = list()
    for train_row in train:
        dist = euclidean_distance(test_row, train_row)
        distances.append((train_row, dist))
    distances.sort(key=lambda tup: tup[1])
    neighbors = list()
    for i in range(num_neighbors):
        neighbors.append(distances[i][0])
    return neighbors
def predict_classification(train, test_row, num_neighbors):
    neighbors = get_neighbors(train, test_row, num_neighbors)
    output_values = [row[-1] for row in neighbors]
    prediction = max(set(output_values), key=output_values.count)
    return prediction

# Test distance function
dataset = [[2.7810836, 2.550537003, 0],
        [1.465489372, 2.362125076, 0],
        [3.396561688, 4.400293529, 0],
        [1.38807019, 1.850220317, 0],
        [3.06407232, 3.005305973, 0],
        [7.627531214, 2.759262235, 1],
        [5.332441248, 2.088626775, 1],
        [6.922596716, 1.77106367, 1],
        [8.675418651, -0.242068655, 1],
        [7.673756466, 3.508563011, 1]]
prediction = predict_classification(dataset, dataset[0], 5)
print('Expected %d, Got %d.' % (dataset[0][-1], prediction))
neighbors = get_neighbors(dataset, dataset[0], 3)
for neighbor in neighbors:
    print(neighbor)
```

Output

```
C:\Users\Student\PycharmProjects\pythonProject\venv\Scripts\python.exe
Expected 0, Got 0.
[2.7810836, 2.550537003, 0]
[3.06407232, 3.005305973, 0]
[1.465489372, 2.362125076, 0]

Process finished with exit code 0
```

Program: 8

Aim:

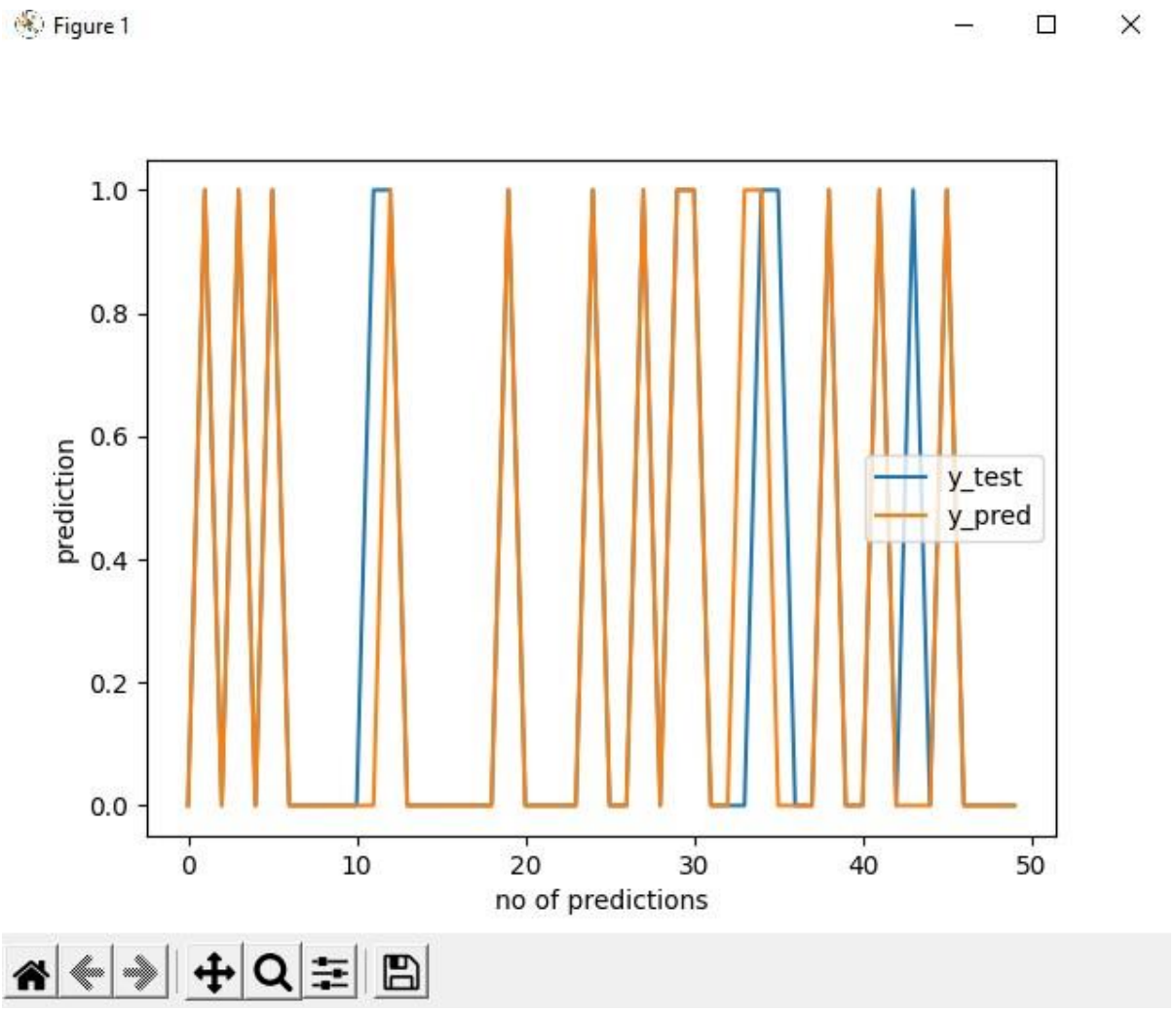
Program to implement Naïve Bayes Algorithm using any standard dataset available in the public domain and find the accuracy of the algorithm.

Program Code:

```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
# Importing the dataset
dataset = pd.read_csv('Social_Network_Ads.csv')
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, -1].values

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 0)

# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
# Training the Naive Bayes model on the Training set
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix, accuracy_score
ac = accuracy_score(y_test,y_pred)
cm = confusion_matrix(y_test, y_pred)

print("Accuracy is:", ac)
print("Confusion Matrix\n",cm)
plt.plot([i for i in range(0, 50)], y_test[20:70])
plt.plot([i for i in range(0, 50)], y_pred[20:70])
plt.xlabel("no of predictions")
plt.ylabel("prediction")
plt.legend(["y_test", "y_pred"])
plt.show()
```

Output

```
C:\Users\Student\PycharmProjects\pythonProject\venv\Scripts\python.exe
Accuracy is: 0.9125
Confusion Matrix
 [[55  3]
 [ 4 18]]

Process finished with exit code 0
```



Program: 9

Aim:

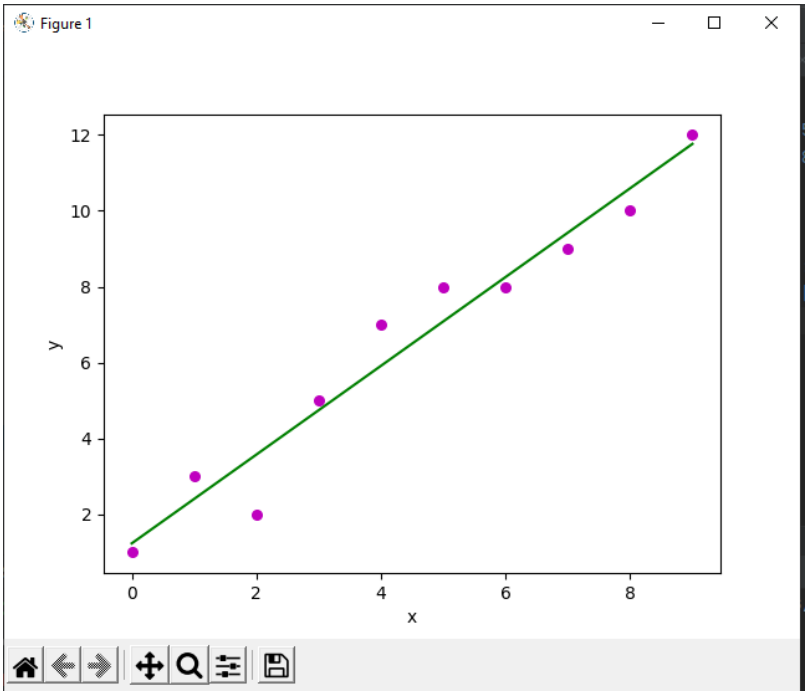
Program to implement linear and multiple regression techniques using any standard dataset available in the public domain and evaluate its performance.

Program Code:

```
import numpy as np
import matplotlib.pyplot as plt


def estimate_coef(x, y):
  # number of observations/points
  n = np.size(x)

  # mean of x and y vector
  m_x = np.mean(x)
  m_y = np.mean(y)

  # calculating cross-deviation and deviation about x
  SS_xy = np.sum(y*x) - n*m_y*m_x
  SS_xx = np.sum(x*x) - n*m_x*m_x

  # calculating regression coefficients
  b_1 = SS_xy / SS_xx
  b_0 = m_y - b_1*m_x

  return (b_0, b_1)

def plot_regression_line(x, y, b):
  # plotting the actual points as scatter plot
  plt.scatter(x, y, color = "m",
       marker = "o", s = 30)
```

```
  # predicted response vector
  y_pred = b[0] + b[1]*x

  # plotting the regression line
  plt.plot(x, y_pred, color = "g")

  # putting labels
  plt.xlabel('x')
  plt.ylabel('y')

  # function to show plot
  plt.show()

def main():
  # observations / data
  x = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
  y = np.array([1, 3, 2, 5, 7, 8, 8, 9, 10, 12])

  # estimating coefficients
  b = estimate_coef(x, y)
  print("Estimated coefficients:\nb_0 = {} \
    \nb_1 = {}".format(b[0], b[1]))

  # plotting regression line
  plot_regression_line(x, y, b)

if__name__== "_main_":
  main()
```

Output



Program: 10

Aim:

Program to implement decision trees using any standard dataset available in the public domain and find the accuracy of the algorithm.

Program Code:

```
import pandas as pd
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

# Function importing Dataset
def importdata():
    balance_data = pd.read_csv(
        'https://archive.ics.uci.edu/ml/machine-learning-' +
        'databases/balance-scale/balance-scale.data',
        sep=',', header=None)

    # Printing the dataset obseravtions
    print("Dataset: ", balance_data.head())
    return balance_data

# Function to split the dataset
def splitdataset(balance_data):
    # Separating the target variable
    X = balance_data.values[:, 1:5]
    Y = balance_data.values[:, 0]

    # Splitting the dataset into train and test
    X_train, X_test, y_train, y_test = train_test_split(
        X, Y, test_size=0.3, random_state=100)

    return X, Y, X_train, X_test, y_train, y_test

# Function to perform training with giniIndex.
def train_using_gini(X_train, X_test, y_train):
    # Creating the classifier object
    clf_gini = DecisionTreeClassifier(criterion="gini",
                                      random_state=100, max_depth=3, min_samples_leaf=5)

    # Performing training
    clf_gini.fit(X_train, y_train)
    return clf_gini

# Function to perform training with entropy.
```

```
def train_using_entropy(X_train, X_test, y_train):
    # Decision tree with entropy
    clf_entropy = DecisionTreeClassifier(
        criterion="entropy", random_state=100,
        max_depth=3, min_samples_leaf=5)

    # Performing training
    clf_entropy.fit(X_train, y_train)
    return clf_entropy

# Function to make predictions
def prediction(X_test, clf_object):
    # Predicton on test with giniIndex
    y_pred = clf_object.predict(X_test)
    print("Predicted values:")
    print(y_pred)
    return y_pred

# Function to calculate accuracy
def cal_accuracy(y_test, y_pred):
     print("Confusion Matrix: ",
         confusion_matrix(y_test, y_pred))

    print("Accuracy : ",
         accuracy_score(y_test, y_pred) * 100)

# Driver code
def main():
    # Building Phase
    data = importdata()
    X, Y, X_train, X_test, y_train, y_test = splitdataset(data)
    clf_gini = train_using_gini(X_train, X_test, y_train)
    clf_entropy = train_using_entropy(X_train, X_test, y_train)

    # Operational  Phase
    print("Results Using Gini Index:")

    # Prediction using gini
    y_pred_gini = prediction(X_test, clf_gini)
    cal_accuracy(y_test, y_pred_gini)

    print("Results Using Entropy:")
    # Prediction using entropy
    y_pred_entropy = prediction(X_test, clf_entropy)
    cal_accuracy(y_test, y_pred_entropy)

# Calling main function
if__name__== "_main_":
    main()
```

Output

```
C:\Users\Student\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/Student/Pycharm
Dataset:     0  1  2  3  4
0  B  1  1  1  1
1  R  1  1  1  2
2  R  1  1  1  3
3  R  1  1  1  4
4  R  1  1  1  5
Results Using Gini Index:
Predicted values:
['R' 'L' 'R' 'R' 'R' 'L' 'R' 'L' 'L' 'L' 'R' 'L' 'L' 'L' 'R' 'L' 'R' 'L'
 'L' 'R' 'L' 'R' 'L' 'L' 'R' 'L' 'L' 'L' 'R' 'L' 'L' 'L' 'R' 'L' 'L' 'L'
 'L' 'R' 'L' 'L' 'R' 'L' 'R' 'L' 'R' 'R' 'L' 'L' 'R' 'L' 'R' 'R' 'L' 'R'
 'R' 'L' 'R' 'R' 'L' 'L' 'R' 'R' 'L' 'L' 'L' 'L' 'L' 'R' 'R' 'L' 'L' 'R'
 'R' 'L' 'R' 'L' 'R' 'R' 'R' 'L' 'R' 'L' 'L' 'L' 'L' 'R' 'R' 'L' 'R' 'L'
 'R' 'R' 'L' 'L' 'L' 'R' 'R' 'L' 'L' 'L' 'R' 'L' 'R' 'R' 'R' 'R' 'R' 'R'
 'R' 'L' 'R' 'L' 'R' 'R' 'L' 'R' 'R' 'R' 'R' 'R' 'L' 'R' 'L' 'L' 'L' 'L'
 'L' 'L' 'L' 'R' 'R' 'R' 'R' 'L' 'R' 'R' 'R' 'L' 'L' 'R' 'L' 'R' 'L' 'R'
 'L' 'L' 'R' 'L' 'L' 'R' 'L' 'R' 'L' 'R' 'R' 'R' 'L' 'R' 'R' 'R' 'R' 'R'
 'L' 'L' 'R' 'R' 'R' 'R' 'L' 'R' 'R' 'R' 'L' 'R' 'L' 'L' 'L' 'L' 'R' 'R'
 'L' 'R' 'R' 'L' 'L' 'R' 'R' 'R']
Confusion Matrix:  [[ 0  6  7]
 [ 0 67 18]
 [ 0 19 71]]
Accuracy :  73.40425531914893
```

```
Results Using Entropy:
Predicted values:
['R' 'L' 'R' 'L' 'R' 'L' 'R' 'L' 'R' 'R' 'R' 'R' 'L' 'L' 'R' 'L' 'R' 'L'
 'L' 'R' 'L' 'R' 'L' 'L' 'R' 'L' 'R' 'L' 'R' 'L' 'R' 'L' 'R' 'L' 'L' 'L'
 'L' 'L' 'R' 'L' 'R' 'L' 'R' 'L' 'R' 'R' 'L' 'L' 'R' 'L' 'L' 'R' 'L' 'L'
 'R' 'L' 'R' 'R' 'L' 'R' 'R' 'R' 'L' 'L' 'R' 'L' 'L' 'R' 'L' 'L' 'L' 'R'
 'R' 'L' 'R' 'L' 'R' 'R' 'R' 'L' 'R' 'L' 'L' 'L' 'L' 'R' 'R' 'L' 'R' 'L'
 'R' 'R' 'L' 'L' 'L' 'R' 'R' 'L' 'L' 'L' 'R' 'L' 'L' 'R' 'R' 'R' 'R' 'R'
 'R' 'L' 'R' 'L' 'R' 'R' 'L' 'R' 'R' 'L' 'R' 'R' 'L' 'R' 'R' 'R' 'L' 'L'
 'L' 'L' 'L' 'R' 'R' 'R' 'R' 'L' 'R' 'R' 'R' 'L' 'L' 'R' 'L' 'R' 'L' 'R'
 'L' 'R' 'R' 'L' 'L' 'R' 'L' 'R' 'R' 'R' 'R' 'R' 'L' 'R' 'R' 'R' 'R' 'R'
 'R' 'L' 'R' 'L' 'R' 'R' 'L' 'R' 'L' 'R' 'L' 'R' 'L' 'L' 'L' 'L' 'L' 'R'
 'R' 'R' 'L' 'L' 'L' 'R' 'R' 'R']
Confusion Matrix:  [[ 0  6  7]
 [ 0 63 22]
 [ 0 20 70]]
Accuracy :  70.74468085106383

Process finished with exit code 0
```

Program: 11

Aim:

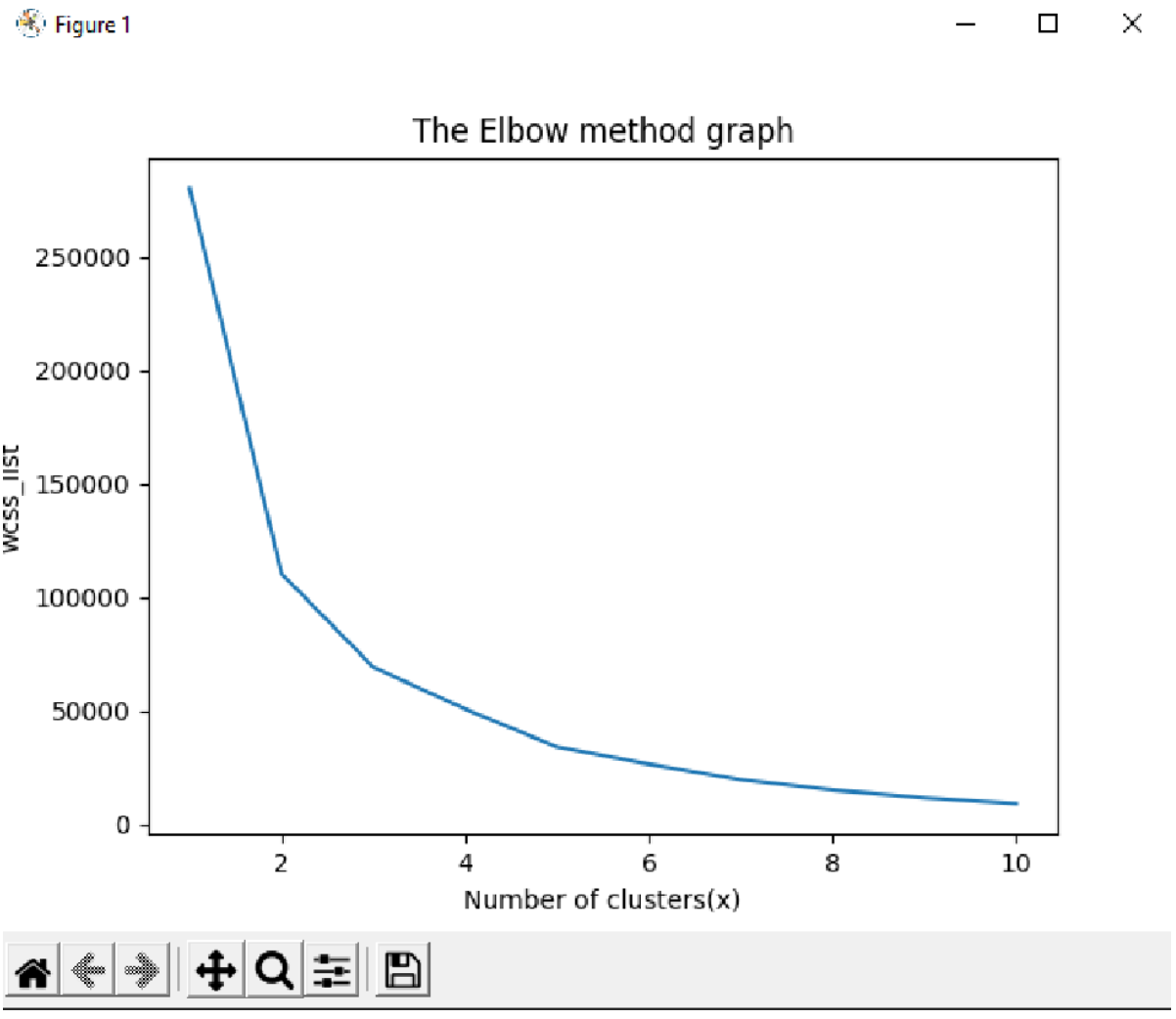
Program to implement k-means clustering technique using any standard dataset available in the public domain

Program Code:

```
import matplotlib.pyplot as mtp
import pandas as pd
from sklearn.cluster import KMeans

dataset = pd.read_csv('world_country_and_usa_states_latitude_and_longitude_values.csv')
x = dataset.iloc[:, [1, 2]].values
print(x)

wcss_list = []

for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state=42)
    kmeans.fit(x)
    wcss_list.append(kmeans.inertia_)
mtp.plot(range(1, 11), wcss_list)
mtp.title('The Elbow method graph')
mtp.xlabel('Number of clusters(x)')
mtp.ylabel('wcss_list')
mtp.show()

kmeans = KMeans(n_clusters=3, init='k-means++', random_state=42)
y_predict = kmeans.fit_predict(x)

print(y_predict)

mtp.scatter(x[y_predict == 0, 0], x[y_predict == 0, 1], s=16, c='blue', label='cluster1')
mtp.scatter(x[y_predict == 1, 0], x[y_predict == 1, 1], s=16, c='green', label='cluster2')
mtp.scatter(x[y_predict == 2, 0], x[y_predict == 2, 1], s=16, c='red', label='cluster3')
mtp.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s=50, c='yellow',
label='Centroids')
mtp.title('Clusters of countries')
mtp.xlabel('Latitude')
mtp.ylabel('Longitude')
mtp.legend()
mtp.show()
```

Output

```
C:\Users\ajcemca\PycharmProjects\newkmeans\venv\Scripts\python.exe C:/Users/ajcemca/PycharmProjects/newkmeans/kmeans.py
[[ 4.25462450e+01  1.60155400e+00]
 [ 2.34240760e+01  5.38478180e+01]
 [ 3.39391100e+01  6.77099530e+01]
 [ 1.70608160e+01 -6.17964280e+01]
 [ 1.82205540e+01 -6.30686150e+01]
 [ 4.11533320e+01  2.01683310e+01]
 [ 4.00690990e+01  4.50381890e+01]
 [ 1.22260790e+01 -6.90600870e+01]
 [-1.12026920e+01  1.78738870e+01]
 [-7.52509730e+01 -7.13890000e-02]
 [-3.84160970e+01 -6.36166720e+01]
 [-1.42709720e+01 -1.70132217e+02]
 [ 4.75162310e+01  1.45500720e+01]
 [-2.52743980e+01  1.33775136e+02]
 [ 1.25211100e+01 -6.99683380e+01]
 [ 4.01431050e+01  4.75769270e+01]
 [ 4.39158860e+01  1.76790760e+01]
 [ 1.31938870e+01 -5.95431980e+01]
 [ 2.36849940e+01  9.03563310e+01]
 [ 5.05038870e+01  4.46993600e+00]
 [ 1.22383330e+01 -1.56159300e+00]
 [ 4.27338830e+01  2.54858300e+01]
 [ 2.59304140e+01  5.06377720e+01]
 [-3.37305600e+00  2.99188860e+01]
```



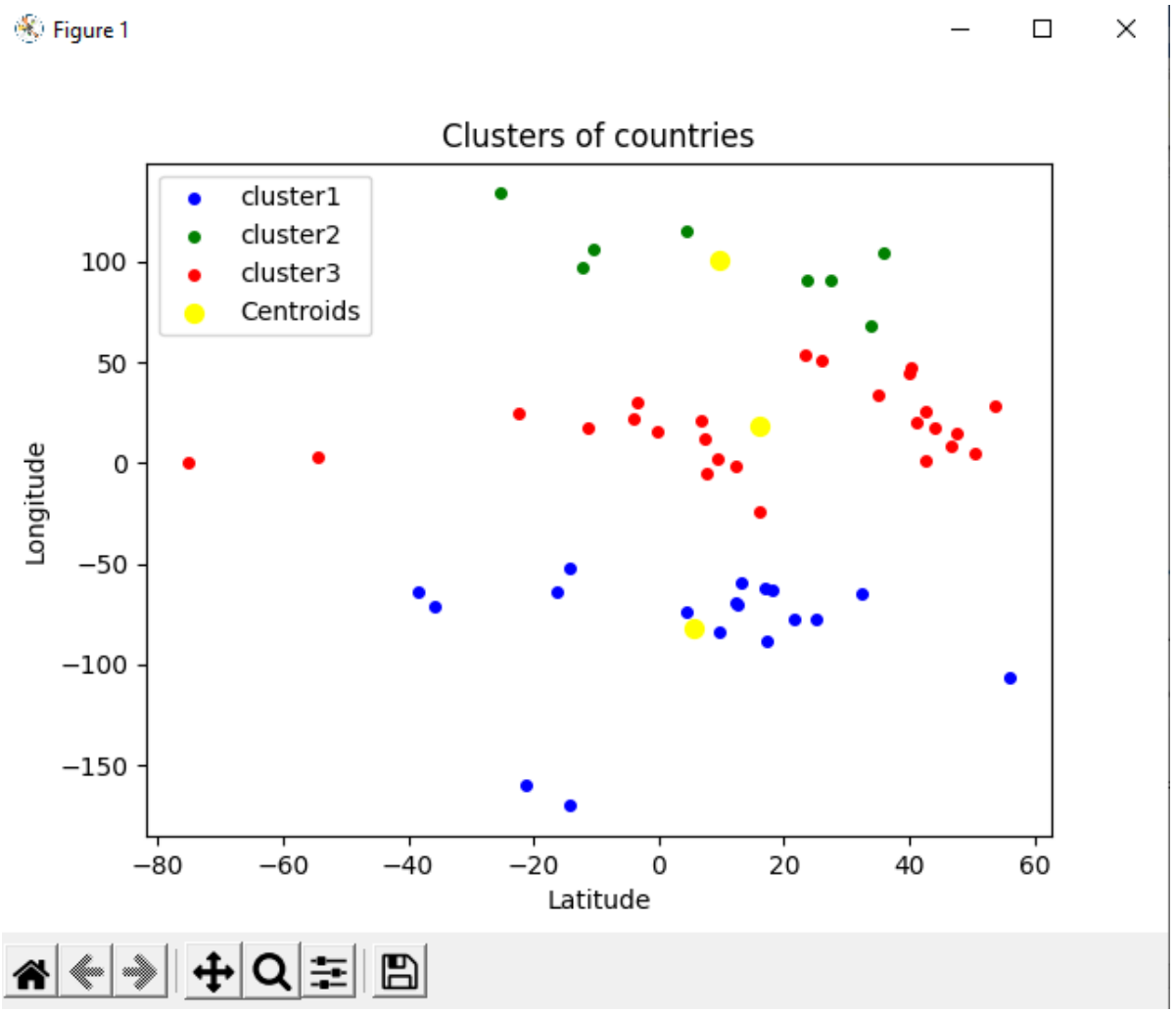


Figure 1
Clusters of countries
cluster1
cluster2
cluster3
Centroids
Longitude
Latitude
100
50
0
−50
−100
−150
−80
−60
−40
−20
0
20
40
60

Program: 12

Aim:

Program to implement simple web crawler using python

Program Code:

```
import requests
import lxml
from bs4 import BeautifulSoup

url = "https://www.rottentomatoes.com/top/bestofrt/"
headers = {
   'User-Agent': 'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/63.0.3239.132 Safari/537.36 OPR/50.0.2762.58 (Edition Yx 01)'}
f = requests.get(url, headers=headers)
movies_lst = []
soup = BeautifulSoup(f.content, 'html.parser')
movies = soup.find('table',
                   {'class': 'table'}).find_all('a')
print("Movies")
print(movies)
num = 0
for anchor in movies:
   urls = 'https://www.rottentomatoes.com' + anchor['href']
   movies_lst.append(urls)
print("Movies List: ")
print(movies_lst)
num +=  1
#movie_url = urls
movie_f = requests.get(urls, headers=headers)
movie_soup = BeautifulSoup(movie_f.content, 'lxml')
movie_content = movie_soup.find('div', {
   'class': 'movie_synopsis clamp clamp-6 js-clamp'
})
print(num, urls, '/n', 'Movie:' + anchor.string.strip())
print("Movie Information: ")
print('Movie info:' + movie_content.string.strip())
```

Output

```
Movies
[<a class="unstyled articleLink" href="/m/it_happened_one_night">
            It Happened One Night (1934)</a>, <a class="unstyled articleLink" href="/m/citizen_kane">
            Citizen Kane (1941)</a>, <a class="unstyled articleLink" href="/m/the_wizard_of_oz_1939">
            The Wizard of Oz (1939)</a>, <a class="unstyled articleLink" href="/m/modern_times">
            Modern Times (1936)</a>, <a class="unstyled articleLink" href="/m/black_panther_2018">
            Black Panther (2018)</a>, <a class="unstyled articleLink" href="/m/parasite_2019">
            Parasite (Gisaengchung) (2019)</a>, <a class="unstyled articleLink" href="/m/avengers_endgame">
            Avengers: Endgame (2019)</a>, <a class="unstyled articleLink" href="/m/1003707-casablanca">
            Casablanca (1942)</a>, <a class="unstyled articleLink" href="/m/knives_out">
            Knives Out (2019)</a>, <a class="unstyled articleLink" href="/m/us_2019">
```

```
Movies List:
['https://www.rottentomatoes.com/m/it_happened_one_night', 'https://www.rottentomatoes.com/m/citizen_kane', 'https://www.ro
1 https://www.rottentomatoes.com/m/the_battle_of_algiers /n Movie:The Battle of Algiers (La Battaglia di Algeri) (1967)
```

```
Movie Information:
Movie info:Paratrooper commander Colonel Mathieu (Jean Martin), a former French Resistance fighter during World War II


Process finished with exit code 0
```

Program: 13

Aim:

Program to implement simple web crawler using python

Program Code:

```
from bs4 import BeautifulSoup
import requests

pages_crawled = []


def crawler(url):
    page = requests.get(url)
    soup = BeautifulSoup(page.text, 'html.parser')
    links = soup.find_all('a')

    for link in links:
        if 'href' in link.attrs:
            if link['href'].startswith('/wiki') and ':' not in link['href']:
                if link['href'] not in pages_crawled:
                    new_link = f"https://en.wikepedia.org{link['href']}"
                    pages_crawled.append(link['href'])
                    try:
                        with open('data.csv', 'a') as file:
                            file.write(f'{soup.title.text}; {soup.h1.text};{link["href"]}\n')
                        crawler(new_link)
                    except:

                        continue


crawler("https://en.wikipedia.org")
```

Output

```
Wikipedia, the free encyclopedia; Main Page;/wiki/Wikipedia
Wikipedia, the free encyclopedia; Main Page;/wiki/Free_content
Wikipedia, the free encyclopedia; Main Page;/wiki/Encyclopedia
Wikipedia, the free encyclopedia; Main Page;/wiki/English_language
Wikipedia, the free encyclopedia; Main Page;/wiki/Brownhills
Wikipedia, the free encyclopedia; Main Page;/wiki/Metropolitan_Borough_of_Walsall
Wikipedia, the free encyclopedia; Main Page;/wiki/Staffordshire
Wikipedia, the free encyclopedia; Main Page;/wiki/Watling_Street
Wikipedia, the free encyclopedia; Main Page;/wiki/Domesday_Book
Wikipedia, the free encyclopedia; Main Page;/wiki/Canals_of_the_United_Kingdom
Wikipedia, the free encyclopedia; Main Page;/wiki/Greed_(game_show)
Wikipedia, the free encyclopedia; Main Page;/wiki/Hector_Waller
Wikipedia, the free encyclopedia; Main Page;/wiki/Ham_House
Wikipedia, the free encyclopedia; Main Page;/wiki/Kobe_Bryant
Wikipedia, the free encyclopedia; Main Page;/wiki/Vanessa_Bryant
Wikipedia, the free encyclopedia; Main Page;/wiki/National_Museum_of_African_American_History_and_Culture
Wikipedia, the free encyclopedia; Main Page;/wiki/Sayfo
Wikipedia, the free encyclopedia; Main Page;/wiki/Seal_Rescue_Ireland
Wikipedia, the free encyclopedia; Main Page;/wiki/Pinniped#Birth_and_parenting
Wikipedia, the free encyclopedia; Main Page;/wiki/Wetsuit
Wikipedia, the free encyclopedia; Main Page;/wiki/Doja_Cat
Wikipedia, the free encyclopedia; Main Page;/wiki/Streets_(song)
Wikipedia, the free encyclopedia; Main Page;/wiki/Billboard_Hot_100
```

Program: 14

Aim:

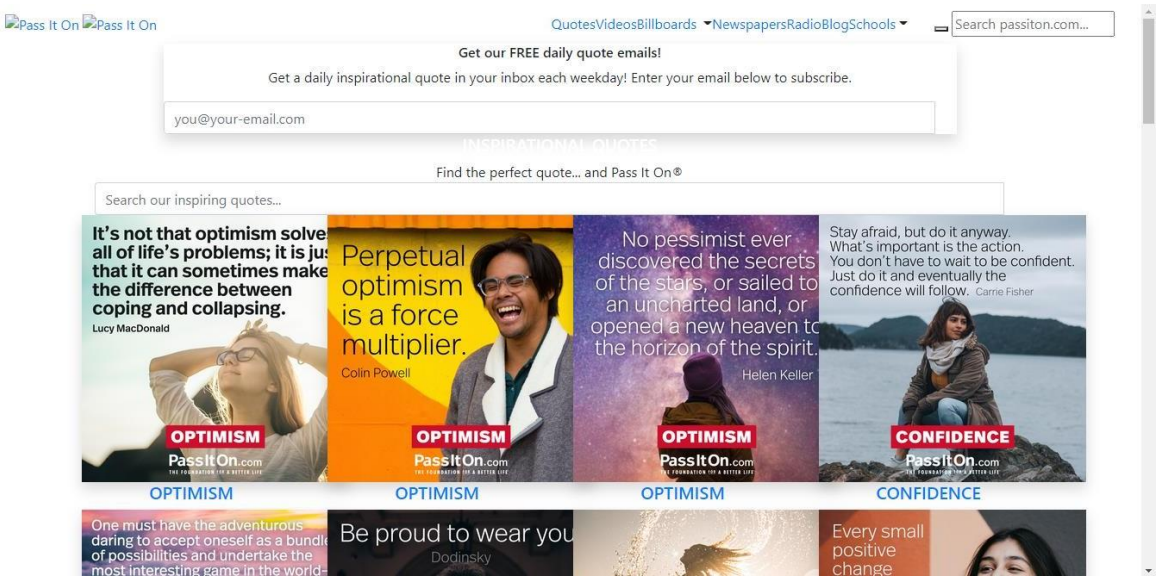
Program to implement scrap of any website

Program Code:

```
import requests
from bs4 import BeautifulSoup
import csv

url = "http://www.values.com/inspirational-quotes"
r = requests.get(url)
print("Content:")
print(r.content)

print("Prettify:")
soup = BeautifulSoup(r.content, 'lxml')
print(soup.prettify())

quotes = []

table = soup.find('div', attrs={'id': 'all_quotes'})

for row in table.find_all('div', attrs={'class': 'col-6 col-lg-3 text-center margin-30px-bottom
sm-margin-30px-top'}):
    quote = {}
    quote['theme'] = row.h5.text
    quote['url'] = row.a['href']
    quote['img'] = row.img['src']
    quote['lines'] = row.img['alt'].split(" #")[0]
    quote['author'] = row.img['alt'].split(" #")[1]
    quotes.append(quote)

filename = 'inspirational_quotion.csv'
with open(filename, 'w', newline='') as f:
    w = csv.DictWriter(f, ['theme', 'url', 'img', 'lines', 'author'])
    w.writeheader()
    for quote in quotes:
        w.writerow(quote)
```

Output

```
C:\Users\mca\PycharmProjectspython\pythonProject2\venv\Scripts\python.exe C:/Users/mca/PycharmProjectspython/pythonProject2/scrap_webpage.py
Content:
b'<!DOCTYPE html>\n<html class="no-js" dir="ltr" lang="en-US">\n    <head>\n        <title>Inspirational Quotes - Motivational Quotes - Leadership Quotes
Prettify:
<!DOCTYPE html>
<html class="no-js" dir="ltr" lang="en-US">
 <head>
  <title>
   Inspirational Quotes - Motivational Quotes - Leadership Quotes | PassItOn.com
  </title>
  <meta charset="utf-8"/>
  <meta content="text/html; charset=utf-8" http-equiv="content-type"/>
  <meta content="IE=edge" http-equiv="X-UA-Compatible"/>
  <meta content="width=device-width,initial-scale=1.0" name="viewport"/>
  <meta content="The Foundation for a Better Life | Pass It On.com" name="description"/>
  <link href="/apple-touch-icon.png" rel="apple-touch-icon" sizes="180x180"/>
  <link href="/favicon-32x32.png" rel="icon" sizes="32x32" type="image/png"/>
  <link href="/favicon-16x16.png" rel="icon" sizes="16x16" type="image/png"/>
  <link href="/site.webmanifest" rel="manifest"/>
  <link color="#c8102e" href="/safari-pinned-tab.svg" rel="mask-icon"/>
  <meta content="#c8102e" name="msapplication-TileColor"/>
  <meta content="#ffffff" name="theme-color"/>
  <link crossorigin="anonymous" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css" integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+
  <link href="/assets/application-2a7a8e6a1c3f620bac9efa86423f5579.css" media="all" rel="stylesheet"/>
  <meta content="authenticity_token" name="csrf-param"/>
  <meta content="/dRa1uuKeod3ZRLd0Bt9aIAKoLWRvWNDzhPDh72mVWq1nGiFCpaHuQVk45eQaZF2u5Re7CixS4xPtfLcgzL4Uw==" name="csrf-token"/>
  <!-- Global site tag (gtag.js) - Google Analytics -->
  <script async="" src="https://www.googletagmanager.com/gtag/js?id=UA-1179005-23">
  </script>
  <script>
```



Program: 15

Aim:

Program for Natural Language Processing which performs n-grams.

Program Code:

```
def generate_ngrams(text, WordsToCombine):
    words = text.split()
    output = []
    for i in range(len(words) - WordsToCombine + 1):
        output.append(words[i:1 + WordsToCombine])
    return output


x = generate_ngrams(text="This is a good book to study", WordsToCombine=3)
print(x)
```

Output

```
C:\Users\Student\PycharmProjects\pythonProject5\venv\Scripts\python.exe C:/Users/Student/PycharmProjects/pythonProject5/02_03(4)py.py
[['This', 'is', 'a', 'good'], ['is', 'a', 'good'], ['a', 'good'], ['good'], []]


Process finished with exit code 0
```

Program: 16

Aim:

Program for Natural Language Processing which performs n-grams (Using inbuilt functions)

Program Code:

```
import nltk
from nltk.util import ngrams

text = "this is a very good book to study"
Ngrams = ngrams(sequence=nltk.wordpunct_tokenize(text), n=3)
for grams in Ngrams:
    print(grams)
```

Output

```
C:\Users\Student\PycharmProjects\pythonProject5\venv\Scripts\python.exe
('this', 'is', 'a')
('is', 'a', 'very')
('a', 'very', 'good')
('very', 'good', 'book')
('good', 'book', 'to')
('book', 'to', 'study')


Process finished with exit code 0
```

Program: 17

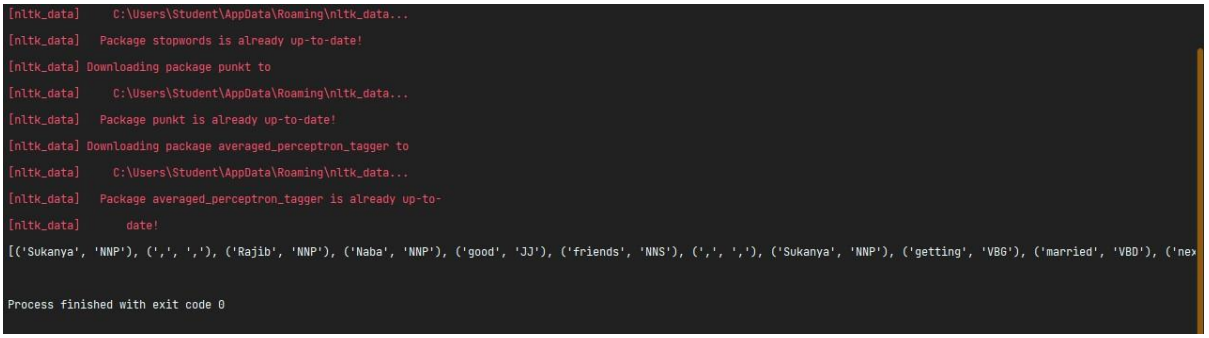
Aim:

Program for Natural Language Processing which performs speech tagging.

Program Code:

```
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize, sent_tokenize
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')

stop_words = set(stopwords.words('english'))
txt = "Sukanya, Rajib and Naba are my good friends,"\
      "Sukanya is getting married next year."\
      "Marriage is a big step in one's life."\
      "It is both exiting and frightening." \
      "But friendship is a sacred bond between people."\
      "It is a special kind of love between us"\
      "Many of you must have tried searching for a friend"\
      "but never found the right one."

tokenized = sent_tokenize(txt)
for i in tokenized:

    wordslist = nltk.word_tokenize(i)


    wordslist = [w for w in wordslist if not w in stop_words]

tagged = nltk.pos_tag(wordslist)

print(tagged)
```

Output


[nltk_data] C:\Users\Student\AppData\Roaming\nltk_data...
[nltk_data] Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data] C:\Users\Student\AppData\Roaming\nltk_data...
[nltk_data] Package punkt is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data] C:\Users\Student\AppData\Roaming\nltk_data...
[nltk_data] Package averaged_perceptron_tagger is already up-to-
[nltk_data] date!
[('Sukanya', 'NNP'), (',', ','), ('Rajib', 'NNP'), ('Naba', 'NNP'), ('good', 'JJ'), ('friends', 'NNS'), (',', ','), ('Sukanya', 'NNP'), ('getting', 'VBG'), ('married', 'VBD'), ('ne
Process finished with exit code 0