**# Program for Intermediate Value Theorem**

**import matplotlib.pyplot as plt**

**import numpy as np**

**#Function Definition**

**def f(x):**

**y=x\*x+5\*x+6**

**return y**

**# Main Program**

**# Input Section**

**a=float(input('Enter initial value of a:'))**

**b=float(input('Enter initial value of b:'))**

**# Plotting function**

**x=[]; y=[]**

**for v in range(0, 11, 1):**

**x.append(a+(b-a)\*v/10)**

**y.append(f(a+(b-a)\*v/10))**

**plt.plot(x, y)**

**plt.grid()**

**plt.show()**

**# Process and output section**

**if f(a)\*f(b)<0:**

**print('root lies in the interval [a, b]=',a,b)**

**elif f(a)\*f(b)==0:**

**print('any one initial value may the root')**

**else:**

**print('No lies in the interval [a, b]=',a,b)**

**# Program for Descartes' Rule of Sign**

**# Input Section**

**order=input("Enter present order from high to low:")**

**coeff=input("Enter corresponding coefficients:")**

**order1=list(map(int,order.split()))**

**coeff1=list(map(int,coeff.split()))**

**print(coeff1)**

**# count of sign changes in f(x)**

**last\_sign = coeff1[0]/abs(coeff1[0])**

**sign\_changes = 0**

**for x in coeff1:**

**sign = x / abs(x)**

**if sign == -last\_sign:**

**sign\_changes = sign\_changes + 1**

**last\_sign = sign**

**p=sign\_changes**

**# Changes f(x) to f(-x)**

**i=0; coeff2=[];**

**for x in order1:**

**if(x%2)==0:**

**coeff2.append(coeff1[i])**

**i=i+1;**

**else:**

**coeff2.append(coeff1[i]\*-1)**

**i=i+1;**

**print(coeff2)**

**# count of sign changes in f(-x)**

**last\_sign = coeff2[0]/abs(coeff2[0])**

**sign\_changes = 0**

**for x in coeff2:**

**sign = x / abs(x)**

**if sign == -last\_sign:**

**sign\_changes = sign\_changes + 1**

**last\_sign = sign**

**q=sign\_changes**

**# Output section**

**print("Number of roots n=",n)**

**print("Number of real positive roots p<=",p)**

**print("Number of real negative roots q<=",q)**

**print("Number of complex roots c<=",n-(p+q))**

**# Program for matrix operations**

**import numpy as np**

**from numpy import linalg as LA**

**# Input section matrix**

**M1 = np.array([[1,4],[5,6]])**

**M2 = np.array([[1,-4],[3,-2]])**

**# Output section matrix**

**# Matrix Addition**

**print("[M1]+[M2]=",M1+M2)**

**# Matrix Subtraction**

**print("[M1]-[M2]=",M1-M2)**

**# Matrix Multiplication**

**print("[M1][M2]=",M1.dot(M2))**

**# Matrix Transpose**

**print("Transpose of [M1]=",M1.transpose())**

**# Matrix Inverse**

**print("Inverse of [M1]=",np.linalg.inv(M1))**

**# Matrix Eigen Values and Vectors**

**w, v = LA.eig(np.array(M1))**

**print("Eigen Values of [M1]=",w)**

**print("Eigen Vectors of [M1]=",v)**