Practice to make a polynmial in numpy.

```
import numpy as np
x=[1,2,3]
np.poly1d(x)
```

```
import numpy as np
x=[1,2,3] # here 1,2,3 are coefficients of the polynomial in descending order
Poly1=np.poly1d(x)
print(Poly1)

Poly2=np.poly1d(x,True) #another format to print polynomial
print(Poly2)
```

```
   2
1 x + 2 x + 3
   3     2
1 x - 6 x + 11 x - 6
```

Code to read data from a CSV file

```
import pandas as pd
import numpy as np

# Read data from CSV file
df = pd.read_csv('data.csv')

# Convert data to numpy arrays
x = df['x values'].values
y = df['y values'].values
```

Function for getting Lagrange Polynmial

```
x = [0, 20,40,60, 80, 100]
y = [26.0, -48.6, 61.6, -71.2, 74.8, -75.2]

import numpy as np
# Function to calculate Lagrange polynomial
def lagrange_poly(x, y):
    n = len(x)
    p = np.poly1d(0.0)
    for i in range(n):
        L = np.poly1d(y[i])
        for j in range(n):
            if j != i:
                L *= np.poly1d([1.0, -x[j]]) / (x[i] - x[j])
        p += L
    return p

# Calculate Lagrange polynomial
p = lagrange_poly(x, y)
print(p)
```

```
            5            4          3         2
-5.329e-06 x + 0.001313 x - 0.1132 x + 3.985 x - 47.81 x + 26
```

For Interpolating at a specific point

```
# Interpolate at a specific point
point = float(input("Enter x-coordinate to interpolate: "))
interp_value = p(point)

# Print Lagrange polynomial and interpolated value
print("Lagrange polynomial is:")
print(p)
print("Interpolated value at x =", point, "is:", interp_value)
```

```
Enter x-coordinate to interpolate: 45
Lagrange polynomial is:
```

```
            5            4          3         2
-5.329e-06 x + 0.001313 x - 0.1132 x + 3.985 x - 47.81 x + 26
Interpolated value at x = 45.0 is: 31.29079589843832
```

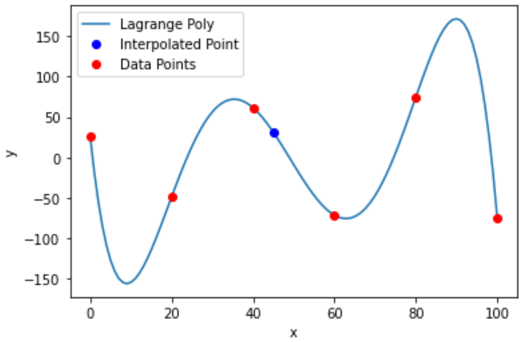
TASK # 01

Solve the above problem manually (Hand written & verify the polynomial & Interpolated value at x = 50 Show all the necessary steps and submitted in the form of PDF along with the project file at GCR

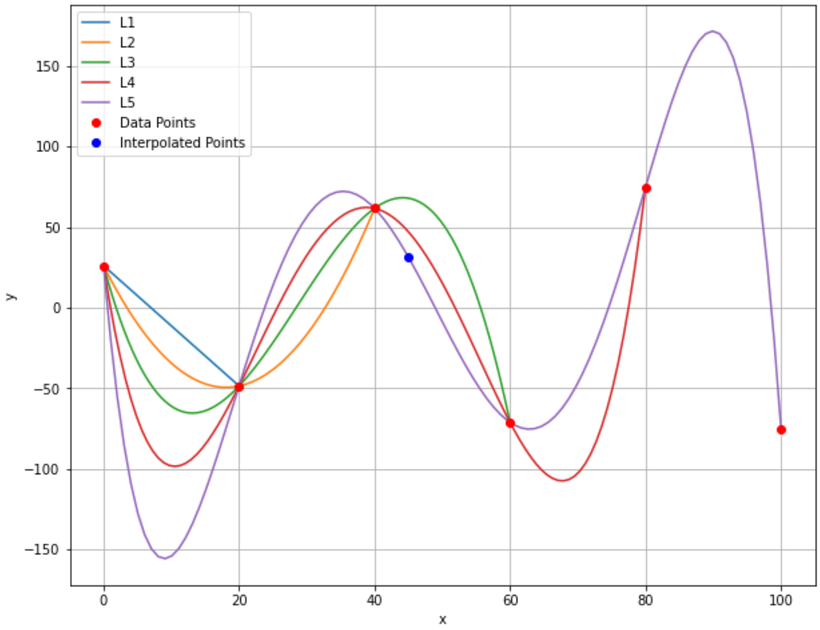
Plotting of Lagrange Polynomial

```
import matplotlib.pyplot as plt
xi=45
yi=31.29079589843832
p = lagrange_poly(x[0:6], y[0:6])
print(p)
xp=np.linspace(0,x[5],100)
yp=p(xp)

plt.plot(xp, yp, label='Lagrange Poly')
plt.plot(xi, yi, 'bo', label='Interpolated Point')
plt.plot(x[0:6], y[0:6], 'ro', label='Data Points')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.show()
```

```
            5            4          3         2
-5.329e-06 x + 0.001313 x - 0.1132 x + 3.985 x - 47.81 x + 26
```




```
fig = plt.figure(figsize = (10,8))
x = [0, 20,40,60, 80, 100]
y = [26.0, -48.6, 61.6,-71.2, 74.8, -75.2]
n=5
for i in range(1,n+1,1):
  p = lagrange_poly(x[0:i+1], y[0:i+1])
  xp=np.linspace(0,x[i],100)
  yp=p(xp)
  plt.plot(xp, yp, label = f"L{i}")
plt.plot(x,y,'ro',label="Data Points")
plt.plot(xi,yi,'bo',label="Interpolated Points")
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.grid()
plt.show()
```

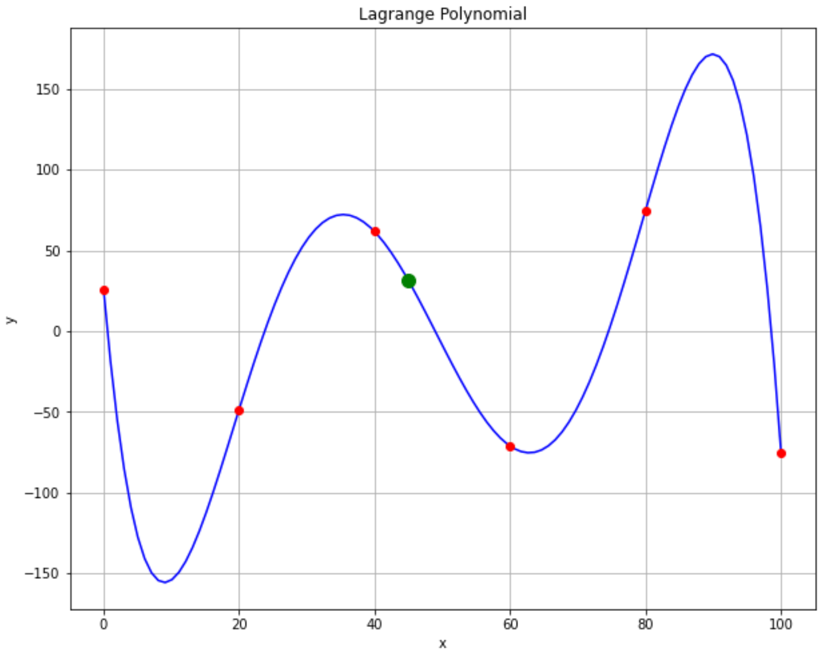


Scipy Implimentation of Lagrange Polynomial

Instead we calculate everything from scratch, in scipy, we can use the lagrange function directly to interpolate the data. Let's see the above example

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import lagrange

# Define the data points
x = np.array([0, 20, 40, 60, 80, 100])
y = np.array([26.0, -48.6, 61.6, -71.2, 74.8, -75.2])

# Define the Lagrange Polynomial
f = lagrange(x, y)

# Find P(50) by evaluating the polynomial at x=50
p_45 = f(45)
print("P(45) =", p_45)

# Plot the Lagrange Polynomial and the data points
x_new = np.linspace(0, 100, 100)
fig = plt.figure(figsize = (10,8))
plt.plot(x_new, f(x_new), 'b', x, y, 'ro')
plt.plot(45, p_45, 'go', markersize=10)
plt.title('Lagrange Polynomial')
plt.grid()
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```

```
P(45) = 31.29079589843832
```



Task # 02

Use the above code and apply the following alteration and show them along with plot

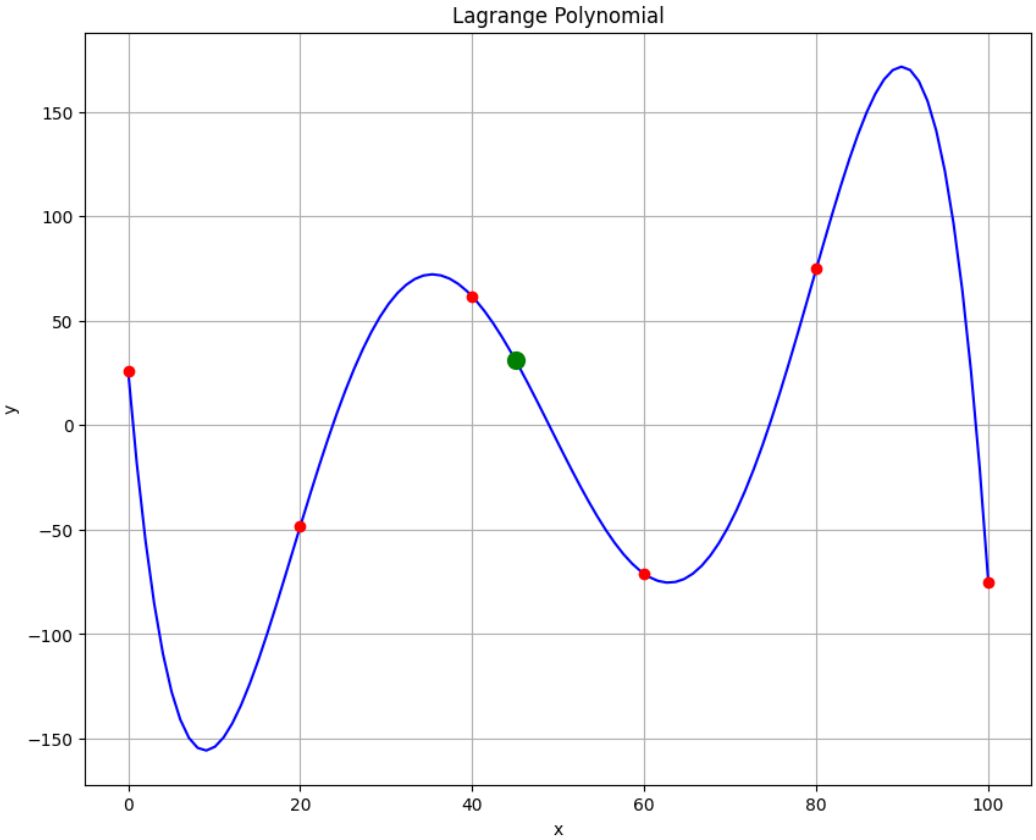
(i) Take input from user and show interpolation at that point along with its plot.

(ii) Also add a code that will display the polynomial too.

Solution of Task # 02

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import lagrange

# Define the data points
x = np.array([0, 20, 40, 60, 80, 100])
y = np.array([26.0, -48.6, 61.6, -71.2, 74.8, -75.2])

# Define the Lagrange Polynomial
f = lagrange(x, y)

# Find P(50) by evaluating the polynomial at x=50
#p_45 = f(45)
#print("P(45) =", p_45)

# Interpolate at a specific point
point = float(input("Enter x-coordinate to interpolate: "))
p = f(point)
print("The value interpolated value is",p )

# Print the polynomial coefficients
print("Lagrange Polynomial:", np.poly1d(f).coefficients)

# Plot the Lagrange Polynomial and the data points
x_new = np.linspace(0, 100, 100)
fig = plt.figure(figsize = (10,8))
plt.plot(x_new, f(x_new), 'b', x, y, 'ro')
plt.plot(45, p, 'go', markersize=10)
plt.title('Lagrange Polynomial')
plt.grid()
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```

```
Enter x-coordinate to interpolate: 45
The value interpolated value is 31.29079589843832
Lagrange Polynomial: [-5.32864583e-06  1.31302083e-03 -1.13188542e-01  3.98529167e+00
 -4.78120000e+01  2.60000000e+01]
```



Code for Newton divided difference Method

```
import numpy as np
def divided_difference_table(x, y):
    n = len(x)
    F = [[0] * n for i in range(n)]
    for i in range(n):
        F[i][0] = y[i]
    for j in range(1, n):
        for i in range(j, n):
            F[i][j] = (F[i][j-1] - F[i-1][j-1]) / (x[i] - x[i-j])
    return F
def newton_div_dif_poly(x,y,xi):
   F=divided_difference_table(x,y) # Saving divided difference in a variable F
   n=len(x)
   prod=np.poly1d(1)
   N=np.poly1d(F[0][0])
   for i in range(1,n):
     prod=np.poly1d(x[0:i],True)
     N+=np.poly1d(F[i][i]*(prod.c))

   print(N)
   print(N(xi))
   return

x = [0, 20,40,60, 80, 100]
y = [26.0, -48.6, 61.6, -71.2, 74.8, -75.2]
newton_div_dif_poly(x, y,45)
```

```
            5            4          3         2
-5.329e-06 x + 0.001313 x - 0.1132 x + 3.985 x - 47.81 x + 26
31.29079589843672
```

Task # 03 (A)

Use the above code by adding divided difference table code i.e. it will show divided difference table.

Task # 03 (B)

With the help of "**pandas**" as shown at the starting of this lab session, read code from provided csv file & Write a code for Newton's divided difference. Print the polynomial and plot the interpolating point too.

Task # 03 (C)

Do part B manually (Mentioned all steps and verify the result.

3 (A) Solution)

```
from tabulate import tabulate
import numpy as np

def divided_difference_table(x, y):
    n = len(x)
    F = [[0] * n for i in range(n)]
    for i in range(n):
        F[i][0] = y[i]
    for j in range(1, n):
        for i in range(j, n):
            F[i][j] = (F[i][j-1] - F[i-1][j-1]) / (x[i] - x[i-j])
    return F

def print_table(x, y):
    F = divided_difference_table(x, y)
    headers = ['x', 'y'] + ['F[{},{}]'.format(i, j) for i in range(1, len(x)) for j in range(i)]
    data = [[x[i], y[i]] + [F[i][j] for j in range(1, i+1)] + [''] * (len(x) - i - 1) for i in range(len(x))]
    print(tabulate(data, headers=headers))

    # Compute the polynomial
    n = len(x)
    prod = np.poly1d(1)
    N = np.poly1d(F[0][0])
    for i in range(1,n):
        prod = np.poly1d(x[0:i], True)
        N += np.poly1d(F[i][i]*(prod.c))
    print('Polynomial: \n', N)

x = [0, 20, 40, 60, 80, 100]
y = [26.0, -48.6, 61.6, -71.2, 74.8, -75.2]
print_table(x, y)
```

```
  x      y  F[1,0]               F[2,0]               F[2,1]                 F[3,0]                  F[3,1]
---  -----  -------------------  -------------------  ---------------------  ----------------------  -----------------------
  0   26
 20  -48.6  -3.7299999999999995
 40   61.6  5.51                 0.23099999999999996
 60  -71.2  -6.640000000000001   -0.30375             -0.008912499999999999
 80   74.8  7.3                  0.34850000000000003  0.010870833333333333   0.00024729166666666666
100  -75.2  -7.5                 -0.37                -0.011975000000000001  -0.0002855729166666667  -5.3286458333333335e-06
Polynomial: 
            5            4          3         2
-5.329e-06 x + 0.001313 x - 0.1132 x + 3.985 x - 47.81 x + 26
```

3 (B)

```
import pandas as pd
from tabulate import tabulate
import numpy as np

# Read X and Y values from CSV file
df = pd.read_csv("interpolation.csv")
x = df["x"].values
y = df["y"].values

# Divided Difference Table
n = len(x)
F = [[0] * n for i in range(n)]
for i in range(n):
    F[i][0] = y[i]
for j in range(1, n):
    for i in range(j, n):
        F[i][j] = (F[i][j-1] - F[i-1][j-1]) / (x[i] - x[i-j])
```

```
# Print the divided difference table using tabulate
headers = ["X"] + ["f[x{0}]".format(i) for i in range(1, n)]
table = []
for i in range(n):
    row = [x[i]] + F[i]
    table.append(row)
print(tabulate(table, headers, floatfmt=".4f"))

# Newton's Divided Difference Polynomial
def newton_div_dif_poly(x,y,xi):
    F = [[0] * n for i in range(n)]
    for i in range(n):
        F[i][0] = y[i]
    for j in range(1, n):
        for i in range(j, n):
            F[i][j] = (F[i][j-1] - F[i-1][j-1]) / (x[i] - x[i-j])

    prod = np.poly1d(1)
    N = np.poly1d(F[0][0])
```