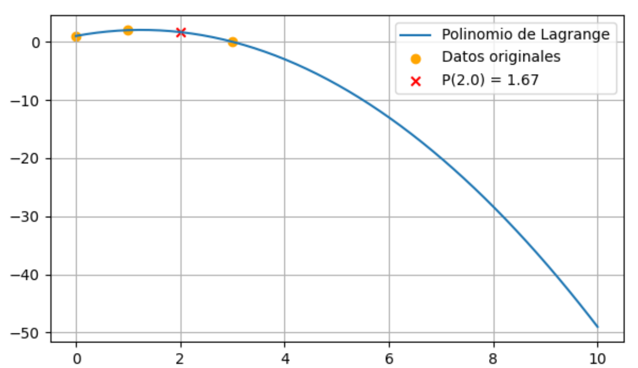
Actividad 1 – II Unidad: Introducción a la aproximación computacional (Interpolación Lagrange)

EJERCICIO 2

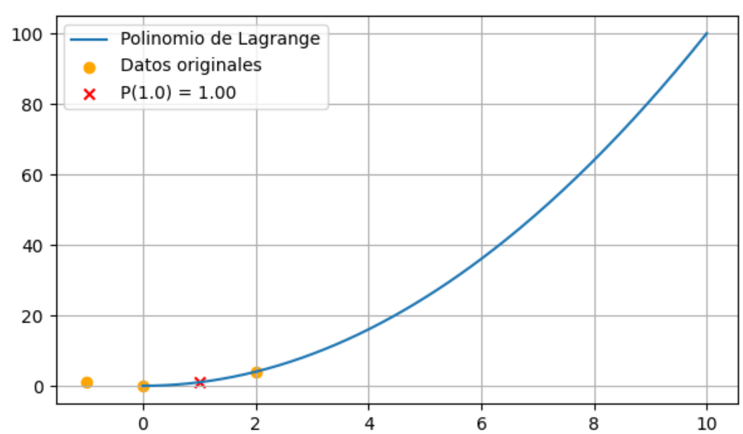
```python
# ============================================
# EJERCICIO 2 - Interpolación de Lagrange
# Puntos: (0,1), (1,2), (3,0)
# ============================================

import numpy as np
import matplotlib.pyplot as plt

# Datos
x_points = np.array([0.0, 1.0, 3.0])
y_points = np.array([1.0, 2.0, 0.0])
x_eval = 2.0

# ---- Función para obtener coeficientes del polinomio de Lagrange ----
def lagrange_poly_coeffs(xs, ys):
    n = len(xs)
    result = np.zeros(n)
    for i in range(n):
        xs_excl = np.delete(xs, i)
        numer_coeffs = np.poly(xs_excl)
        denom = np.prod(xs[i] - xs_excl)
        if len(numer_coeffs) < len(result):
            numer_coeffs = np.pad(numer_coeffs, (len(result) - len(numer_coeffs), 0), 'constant')
        result = result + ys[i] * numer_coeffs / denom
    return result
# ---- Construir polinomio y evaluar ----
coeffs = lagrange_poly_coeffs(x_points, y_points)
P = np.poly1d(coeffs)
y_eval = P(x_eval)
print("=== EJERCICIO 2 ===")
print("Polinomio interpolador P(x):")
print(P)
print(f"P({x_eval}) = {y_eval:.6f}")

# ---- Graficar ----
x_plot = np.linspace(0, 10, 200)
y_plot = P(x_plot)

plt.figure(figsize=(7,4))
plt.plot(x_plot, y_plot, label='Polinomio de Lagrange')
plt.scatter(x_points, y_points, color='orange', label='Datos originales')
plt.scatter([x_eval], [y_eval], color='red', marker='x', label=f'P({x_eval}) = {y_eval:.2f}')
plt.grid(True)
plt.legend()
plt.show()
```



Grafica Interpretación: Se puede observar en la gráfica como el polinomio primero crece y luego disminuye esto debido a que el polinomio cuadrático no presenta una relación lineal respecto a X y Y.

EJERCICIO 3

```python
# ==============================================
# EJERCICIO 3 - Interpolación de Lagrange
# Puntos: (-1,1), (0,0), (2,4)
# ==============================================
import numpy as np
import matplotlib.pyplot as plt

# Datos
x_points = np.array([-1.0, 0.0, 2.0])
y_points = np.array([1.0, 0.0, 4.0])
x_eval = 1.0

# ---- Función para obtener coeficientes del polinomio de Lagrange ----
def lagrange_poly_coeffs(xs, ys):
    n = len(xs)
    result = np.zeros(n)
    for i in range(n):
        xs_excl = np.delete(xs, i)
        numer_coeffs = np.poly(xs_excl)
        denom = np.prod(xs[i] - xs_excl)
        if len(numer_coeffs) < len(result):
            numer_coeffs = np.pad(numer_coeffs, (len(result) - len(numer_coeffs), 0), 'constant')
        result = result + ys[i] * numer_coeffs / denom
    return result

# ---- Construir polinomio y evaluar ----
coeffs = lagrange_poly_coeffs(x_points, y_points)
P = np.poly1d(coeffs)
y_eval = P(x_eval)

print("=== EJERCICIO 3 ===")
print("Polinomio interpolador P(x):")
print(P)
print(f"P({x_eval}) = {y_eval:.6f}")
# ---- Graficar ----
x_plot = np.linspace(0, 10, 200)
y_plot = P(x_plot)

plt.figure(figsize=(7,4))
plt.plot(x_plot, y_plot, label='Polinomio de Lagrange')
plt.scatter(x_points, y_points, color='orange', label='Datos originales')
plt.scatter([x_eval], [y_eval], color='red', marker='x', label=f'P({x_eval}) = {y_eval:.2f}')
plt.grid(True)
plt.legend()
plt.show()
```



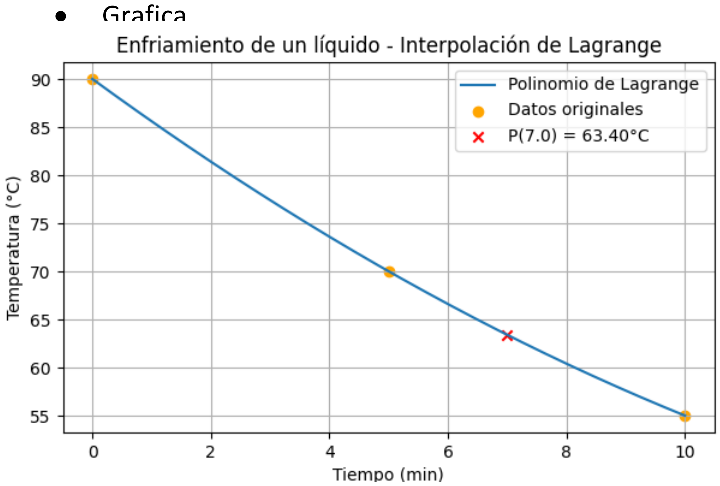
Interpretación: La gráfica muestra un comportamiento **parabólico ascendente**, característico de procesos donde la variable dependiente crece aceleradamente con el tiempo o el valor de entrada, como una función de energía o distancia cuadrática.

EJERCICIO 5:

- Programa en python

```python
# EJERCICIO 5 - Interpolación de Lagrange
# Enfriamiento de un líquido
# Puntos: (0,90), (5,70), (10,55)
import numpy as np
import matplotlib.pyplot as plt
# ---- Datos ----
x_points = np.array([0.0, 5.0, 10.0])     # Tiempo (min)
y_points = np.array([90.0, 70.0, 55.0])   # Temperatura (°C)
x_eval = 7.0                              # Evaluar a los 7 minutos

# ---- Función para obtener coeficientes del polinomio de Lagrange ----
def lagrange_poly_coeffs(xs, ys):
    n = len(xs)
    result = np.zeros(n)
    for i in range(n):
        xs_excl = np.delete(xs, i)
        numer_coeffs = np.poly(xs_excl)
        denom = np.prod(xs[i] - xs_excl)
        if len(numer_coeffs) < len(result):
            numer_coeffs = np.pad(numer_coeffs, (len(result) - len(numer_coeffs), 0), 'constant')
        result = result + ys[i] * numer_coeffs / denom
    return result
# ---- Construir polinomio y evaluar ----
coeffs = lagrange_poly_coeffs(x_points, y_points)
P = np.poly1d(coeffs)
y_eval = P(x_eval)

print("=== EJERCICIO 5 ===")
print("Polinomio interpolador P(x):")
print(P)
print(f"P({x_eval}) = {y_eval:.6f} °C")

# ---- Graficar ----
x_plot = np.linspace(0, 10, 200)
y_plot = P(x_plot)

plt.figure(figsize=(7,4))
plt.plot(x_plot, y_plot, label='Polinomio de Lagrange')
plt.scatter(x_points, y_points, color='orange', label='Datos originales')
plt.scatter([x_eval], [y_eval], color='red', marker='x', label=f'P({x_eval}) = {y_eval:.2f}°C')
plt.title("Enfriamiento de un líquido - Interpolación de Lagrange")
plt.xlabel("Tiempo (min)")
plt.ylabel("Temperatura (°C)")
plt.grid(True)
plt.legend()
plt.show()
```

- Grafica



- Interpretacion: La gráfica modela el enfriamiento de un líquido (por ejemplo, agua caliente) al paso del tiempo.

Al principio, la temperatura baja rápidamente (de 90 °C a 70 °C en los primeros 5 min).

Luego, la curva se aplana, mostrando que la pérdida de calor es más lenta conforme el líquido se acerca a la temperatura ambiente.

- Cálculo matemático

$X_0 = 0 \quad Y_0 = 90 \quad X_1 = 5 \quad Y_1 = 70 \quad X_2 = 10 \quad Y_2 = 55$

$$L_0(x) = \frac{(x-5)(x-10)}{(0-5)(0-10)} = L_0(x) = \frac{x^2-15x+50}{50}$$

$$L_1(x) = \frac{(x-0)(x-10)}{(5-0)(5-10)} = L_1(x) = \frac{x^2-10x}{-25} = \frac{x^2-10x}{25}$$

$$L_2(x) = \frac{(x-0)(x-5)}{(10-0)(10-5)} = \frac{x^2-5x}{50}$$

$$P(x) = 90L_0 + 70L_1 + 55L_2$$

$$P(x) = \frac{1}{10}x^2 - \frac{9}{2}x + 90$$

$$P(7) = \frac{1}{10} \cdot 49 - \frac{9}{2} \cdot 7 + 90 = 4.9 - 31.5 + 90 = 63.4°C$$