**UNIVERSIDAD MARIANO GALVEZ DE GUATEMALA**

**“Proyecto de Algebra Lineal”**

**Lenguaje de Python**

**En la presente se detalla el enlace de GitHub**

**Integrantes:**

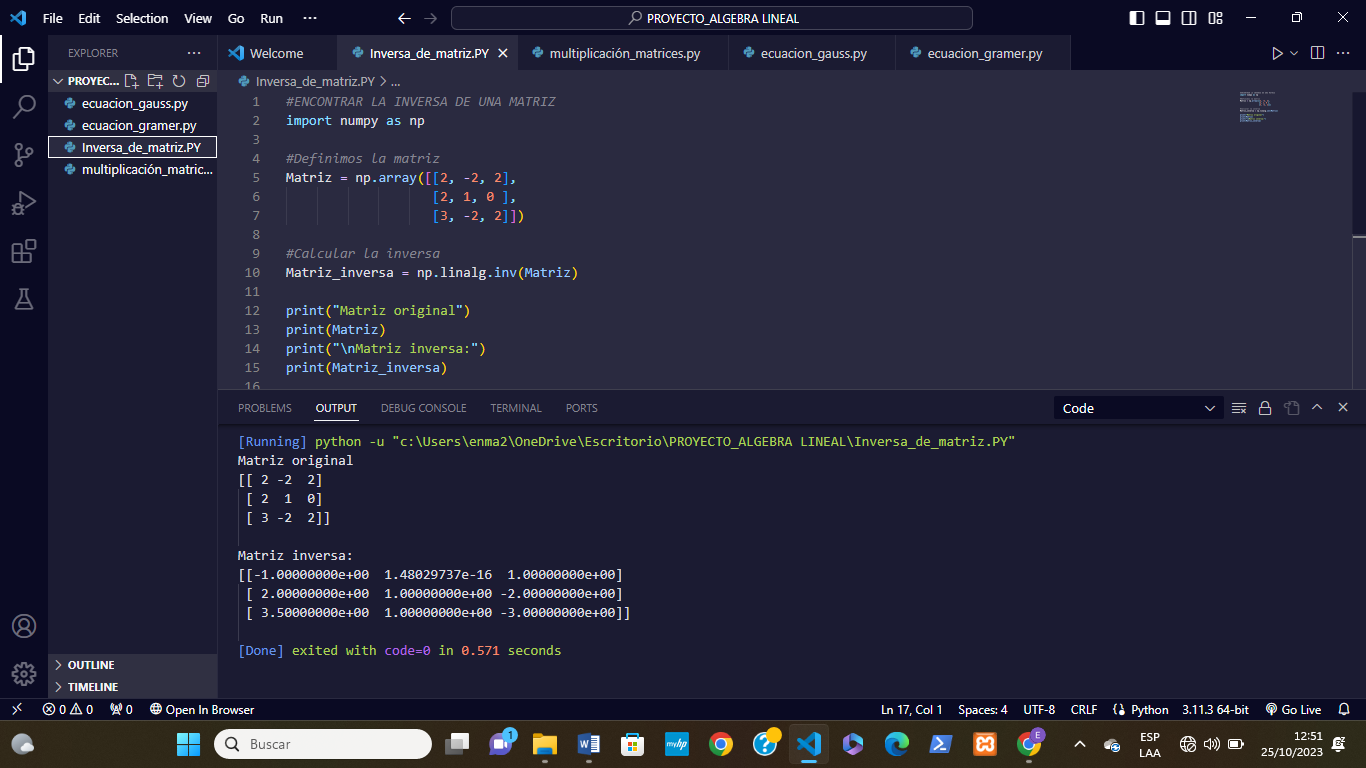
* **Keily Atalia López Hernández**
* **Eugenia del rosario López Nájera**
* **Enma Leticia Ramírez castro**

**Curso: Algebra lineal**

**Licenciado:**

**2,023**

1. ENCONTRAR LA INVERSA DE UNA MATRIZ
2. *#ENCONTRAR LA INVERSA DE UNA MATRIZ*
3. import numpy as np
4. *#Definimos la matriz*
5. Matriz = np.array([[2, -2, 2],
6. [2, 1, 0 ],
7. [3, -2, 2]])
8. *#Calcular la inversa*
9. Matriz\_inversa = np.linalg.inv(Matriz)
10. print("Matriz original")
11. print(Matriz)
12. print("\nMatriz inversa:")
13. print(Matriz\_inversa)



2. MULTIPLICACIÓN DE MATRICES

*#mULTIPLICASION DE MATRIZ*

def multiplicacion\_matrices(matriz1, matriz2):

    if len(matriz1[0])!= len(matriz2):

        return

    resultado = [[0 for \_ in range(len(matriz2[0]))] for \_ in range(len(matriz1))]

    for i in range(len(matriz1)):

        for j in range(len(matriz2[0])):

                       for k in range(len(matriz2)):

                           resultado[i][j] += matriz1[i][k]\* matriz2[k][j]

    return resultado

*#describimos las matrices*

matriz1 = ([[1, 6],

           [0, 4],

           [-2, 3]])

matriz2 = ([[7, 1, 4],

           [2, -3, 5]])

resultado = multiplicacion\_matrices(matriz1, matriz2)

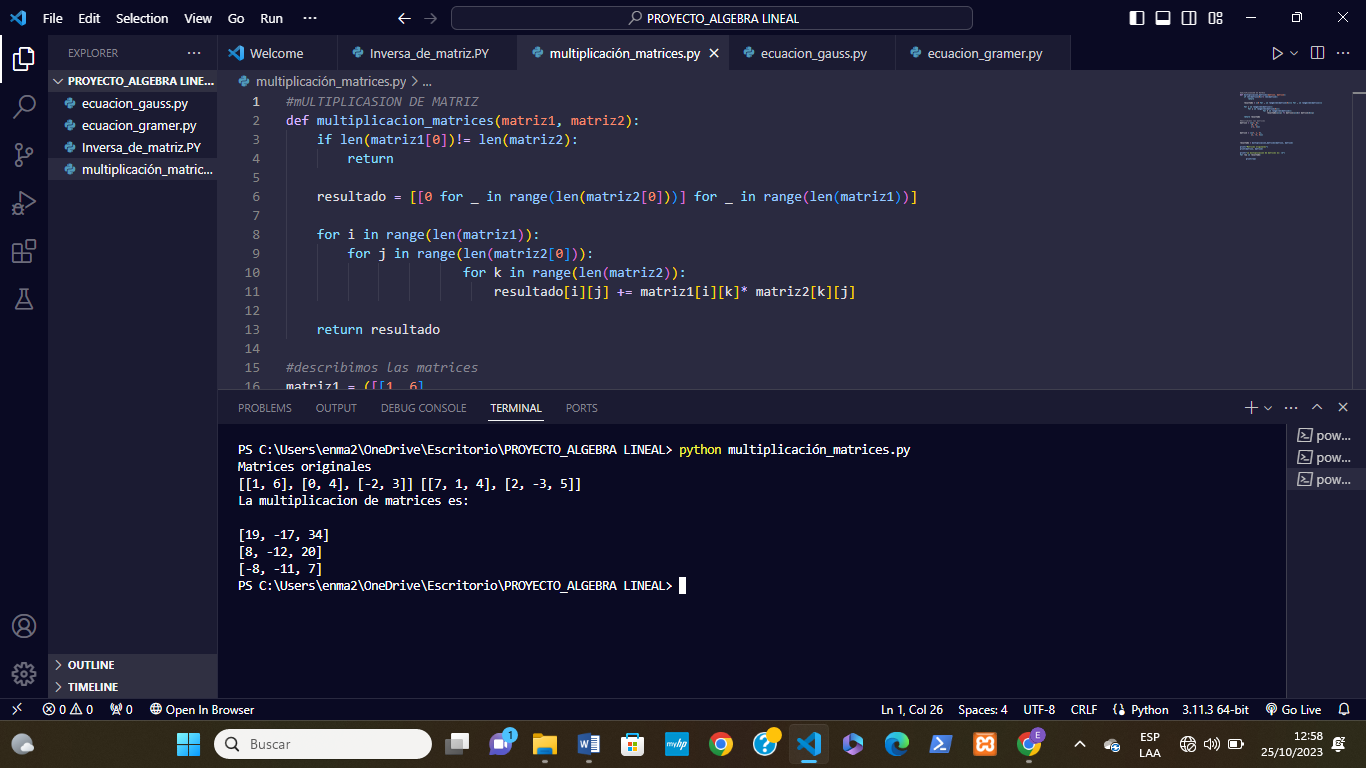
print("Matrices originales")

print(matriz1, matriz2)

print("La multiplicacion de matrices es: \n")

for row in resultado:

      print(row)



3. RESOLVER SISTEMAS DE ECUACIONES LINEALES (2X2) REGLA DE GAUSS JORDAN

*#SISTEMA DE ECUACION LINEAL USANDO LA REGLA DE GAUSS JORDAN*

import matplotlib.pyplot as plt

import numpy as np

*#sistema de ecucaiones definido*

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

B = np.array([5, 2])

*#Definimos la variable x*

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

*#Graficamos las ecuaciones a resolver.*

X\_vals = np.linspace(-10, 10, 100)

Y1 = (5 - 2 \* X\_vals) / 1

Y2 = (X\_vals -2) / -3

plt.figure(figsize=(6, 5))

plt.plot(X\_vals,Y1, label='2X + y = 5')

plt.plot(X\_vals,Y2, label='X - 3y = 2')

plt.xlim(-10, 10)

plt.ylim(-10,10)

*#Solucion grafica*

plt.scatter(X[0], X[1], color="red", label='solucion (X, Y)')

plt.xlabel('X')

plt.xlabel('Y')

plt.axhline(0, color='black',linewidth=0.5)

plt.axvline(0, color='black',linewidth=0.5)

plt.grid(color = 'gray', linestyle = '--', linewidth=0.5)

plt.legend()

plt.title('SISTEMA DE ECUACIONES LINEALES DE GAUSS JORDAN DE 2\*2')

plt.show()

print(f"La solución es X = {X[0]}, Y = {X[1]}")



3. RESOLVER SISTEMAS DE ECUACIONES LINEALES (3X3) REGLA DE CRAMER

*#SISTEMA DE ECUACION LINEAL USANDO LA REGLA DE GRAMER*

import matplotlib.pyplot as plt

import numpy as np

*#sistema de ecucaiones definido*

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

              [3, -2, -3],

              [8, 2, 5]])

B = np.array([6, 5, 11])

*#Funcion con regla de cramer*

def cramer\_rule(A, B):

    det\_A = np.linalg.det(A)

    X = []

    for i in range(A.shape[1]):

        A\_temp = A.copy()

        A\_temp[:, i] = B

        det\_A\_temp = np.linalg.det(A\_temp)

        X.append(det\_A\_temp / det\_A)

    return X

*#aca se resuelve el sistema*

solution = cramer\_rule(A, B)

for i, sol  in enumerate(solution):

    print(f'x{i+1} = {sol}')

*#funcion de grafica*

fig = plt.figure()

ax = fig.add\_subplot(111, projection='3d')

ax.quiver(0, 0, 0, solution[0], solution[1], solution[2], color='m')

ax.set\_xlim(0, solution[0])

ax.set\_ylim(0, solution[1])

ax.set\_zlim(0, solution[2])

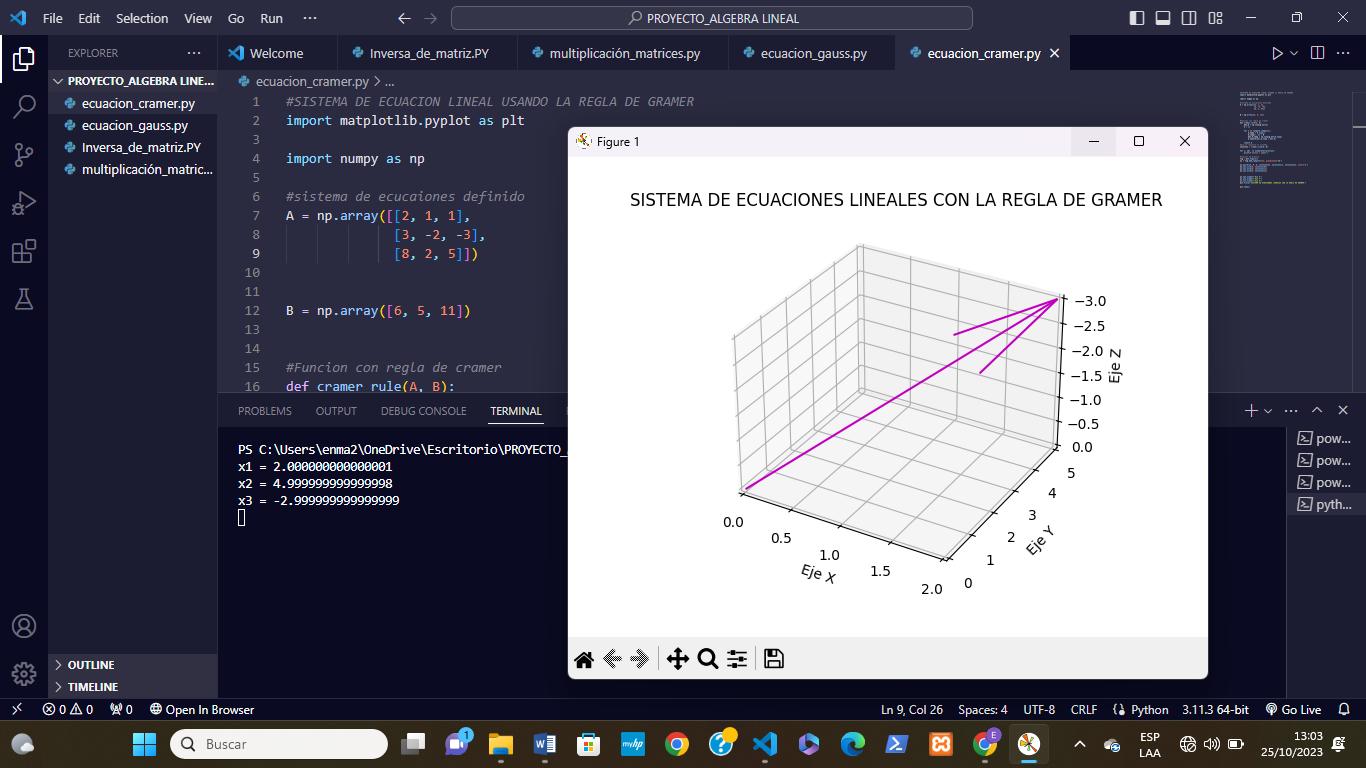
ax.set\_xlabel('Eje X')

ax.set\_ylabel('Eje Y')

ax.set\_zlabel('Eje Z')

plt.title('SISTEMA DE ECUACIONES LINEALES CON LA REGLA DE GRAMER')

plt.show()

****