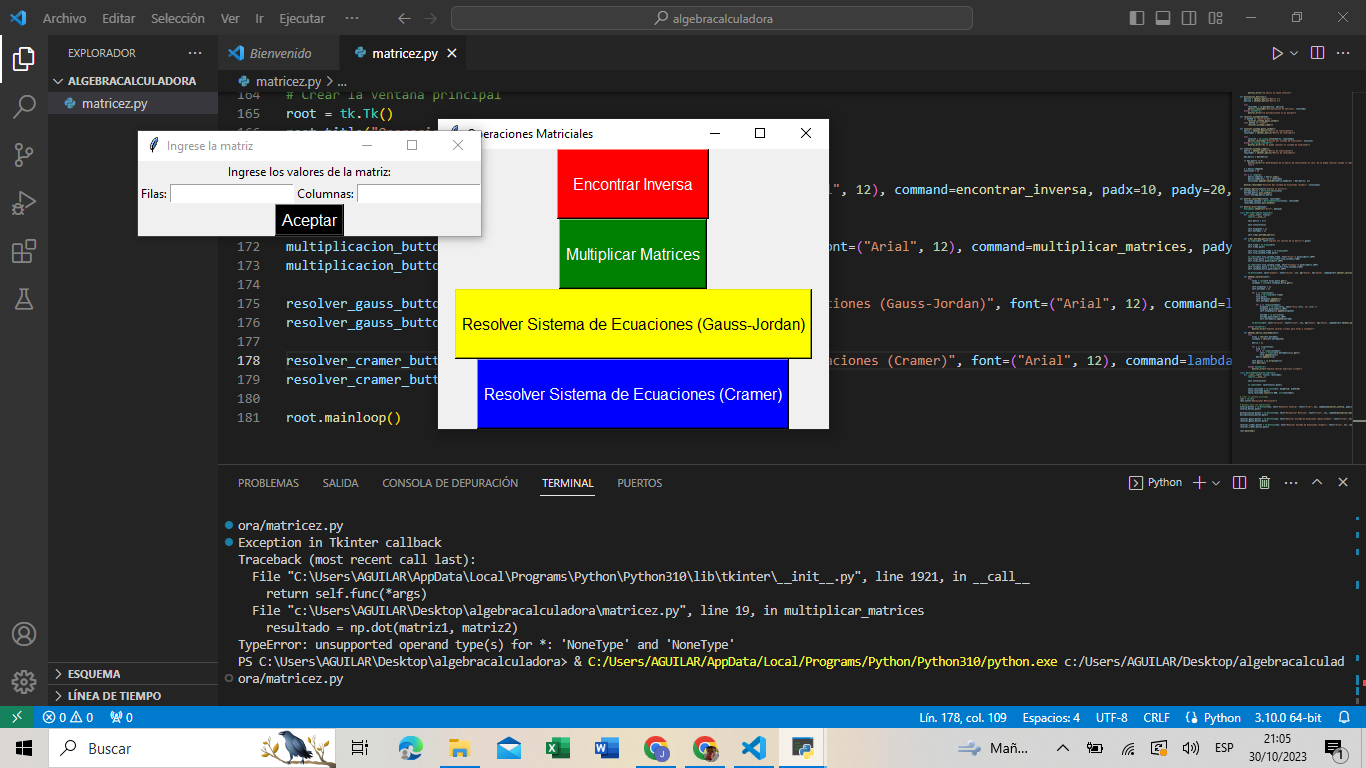
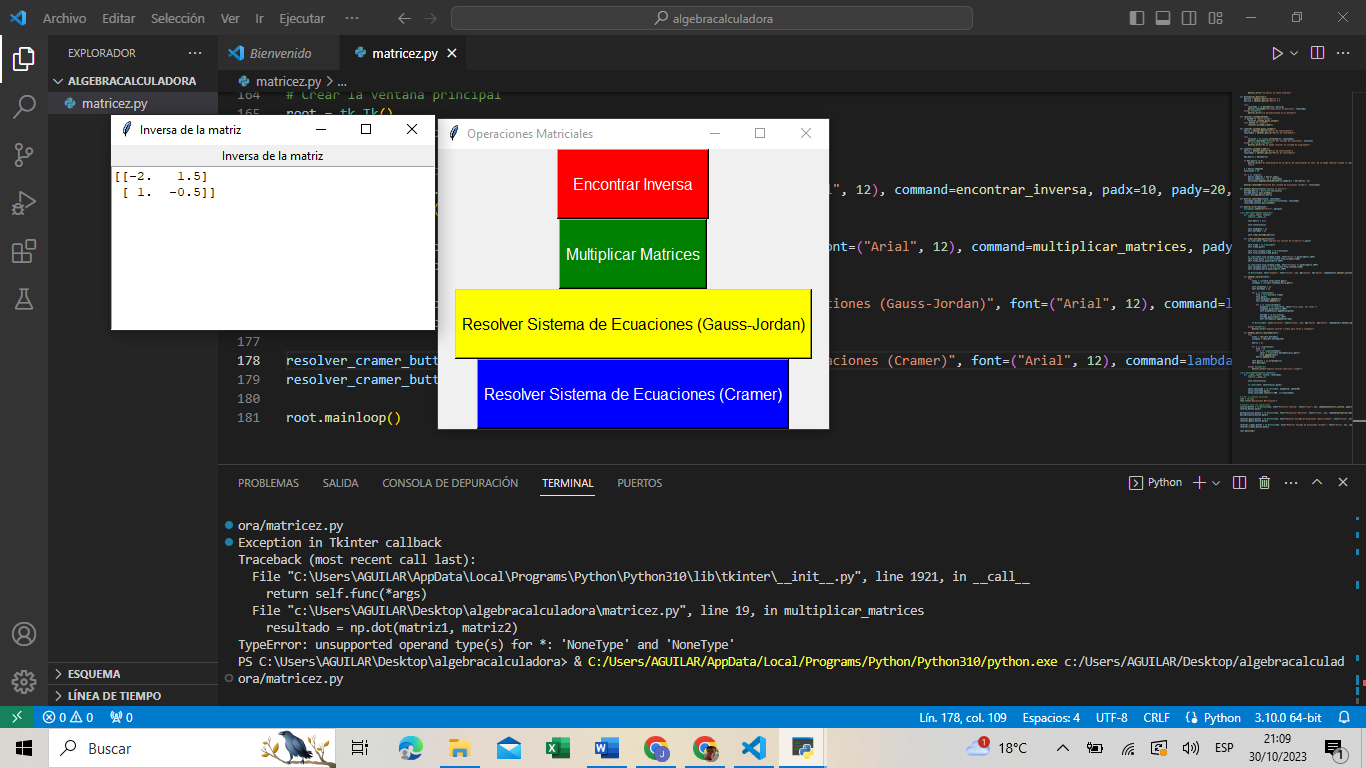
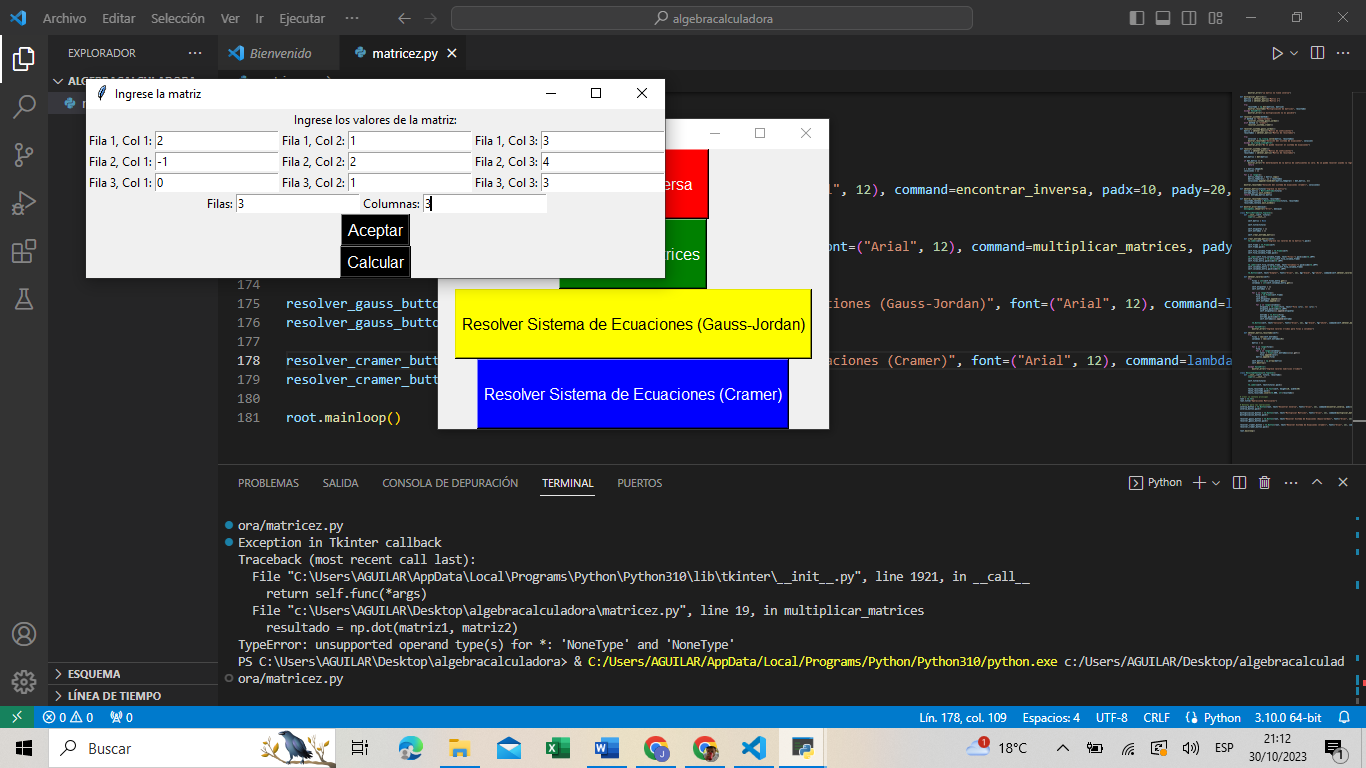
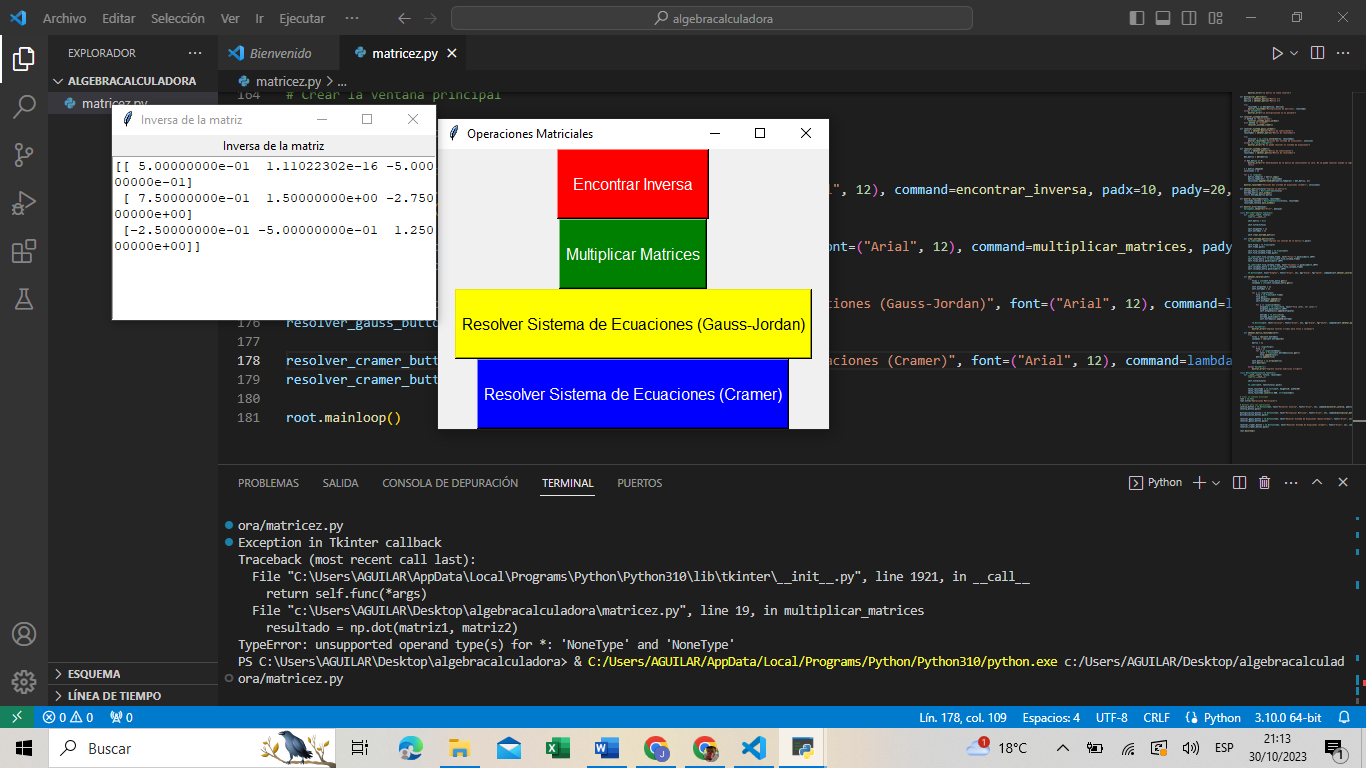
INVERSA 2x2





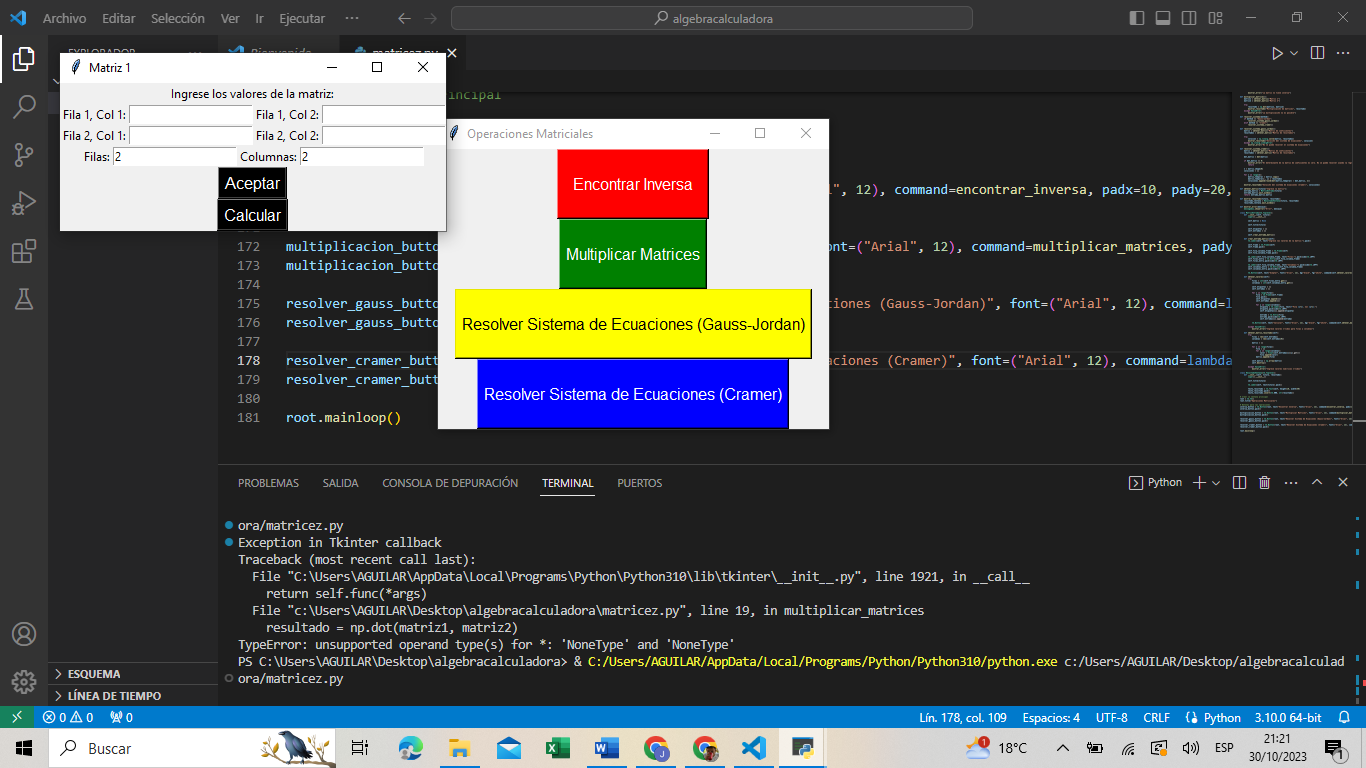
INVERSA 3x3

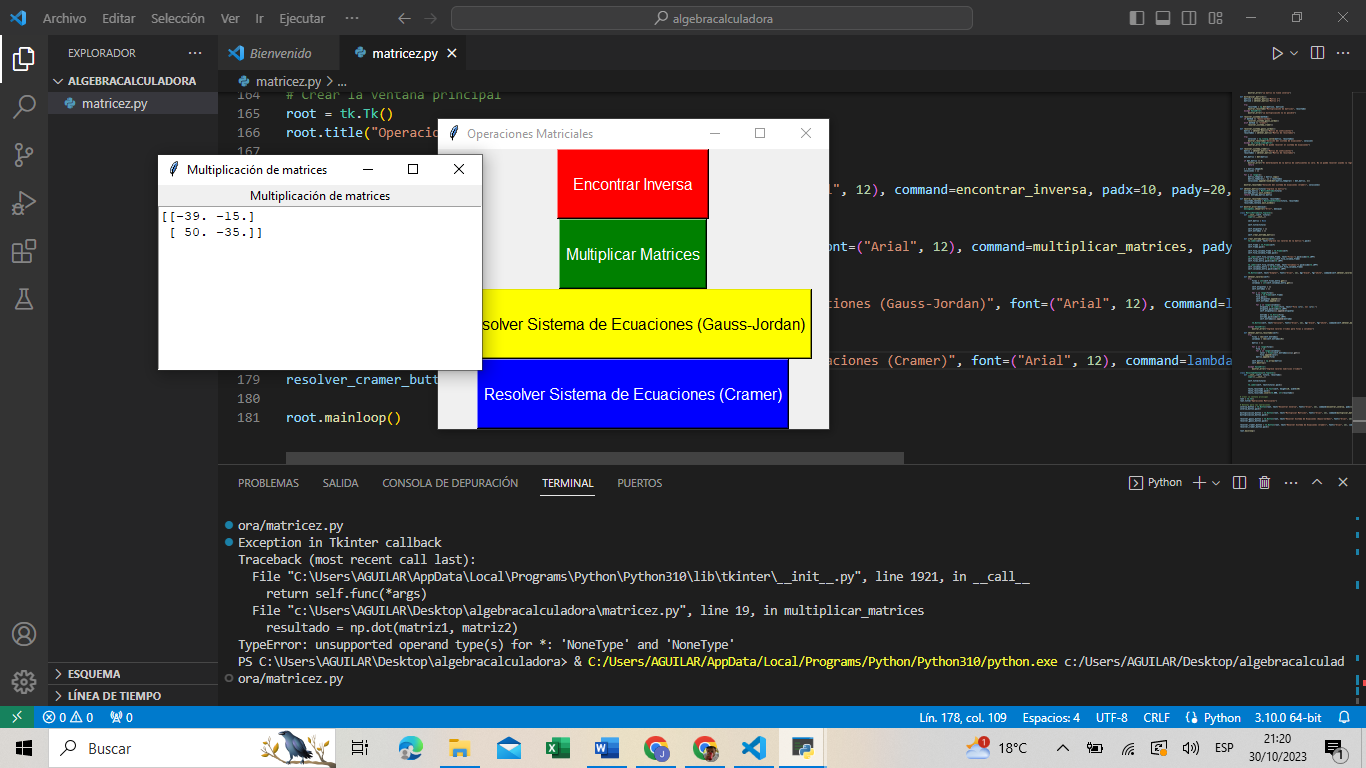




MULTIPLICACION DE MATRICES 2x2

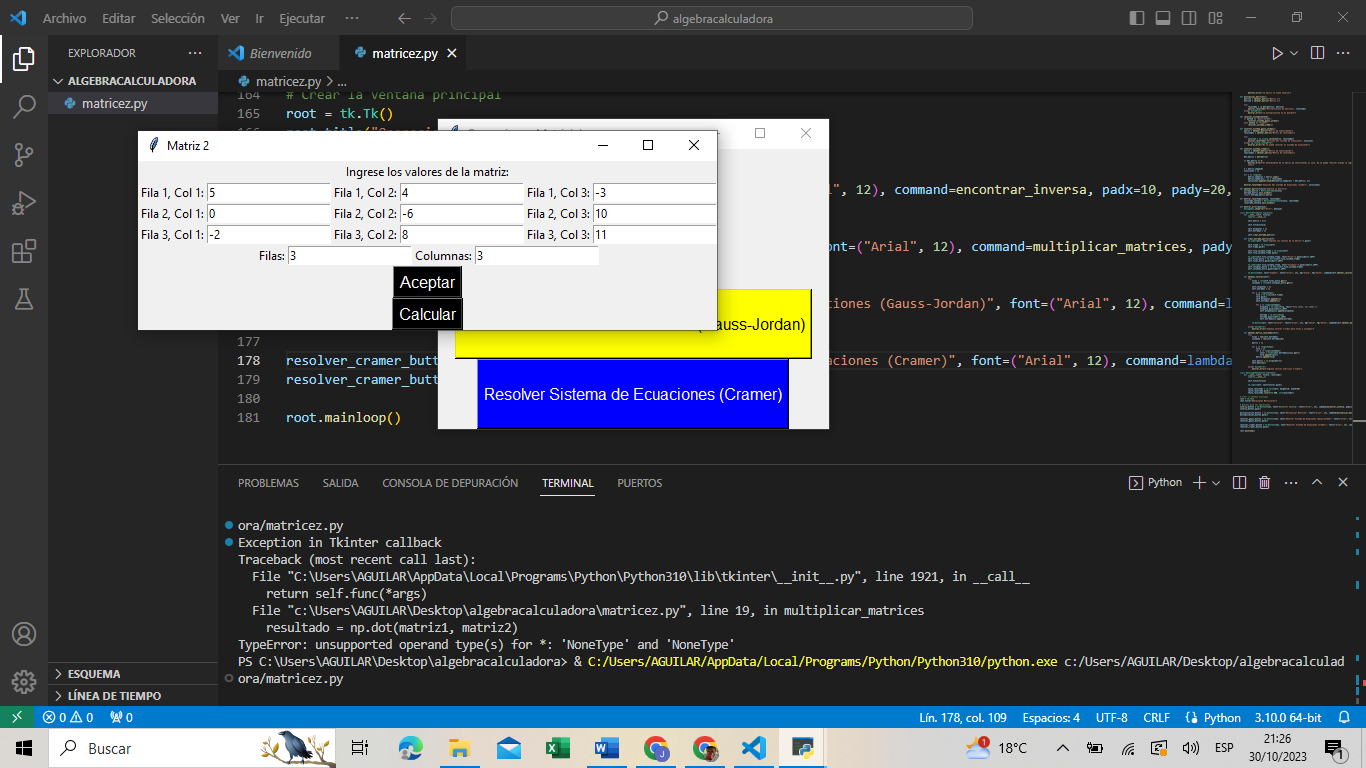
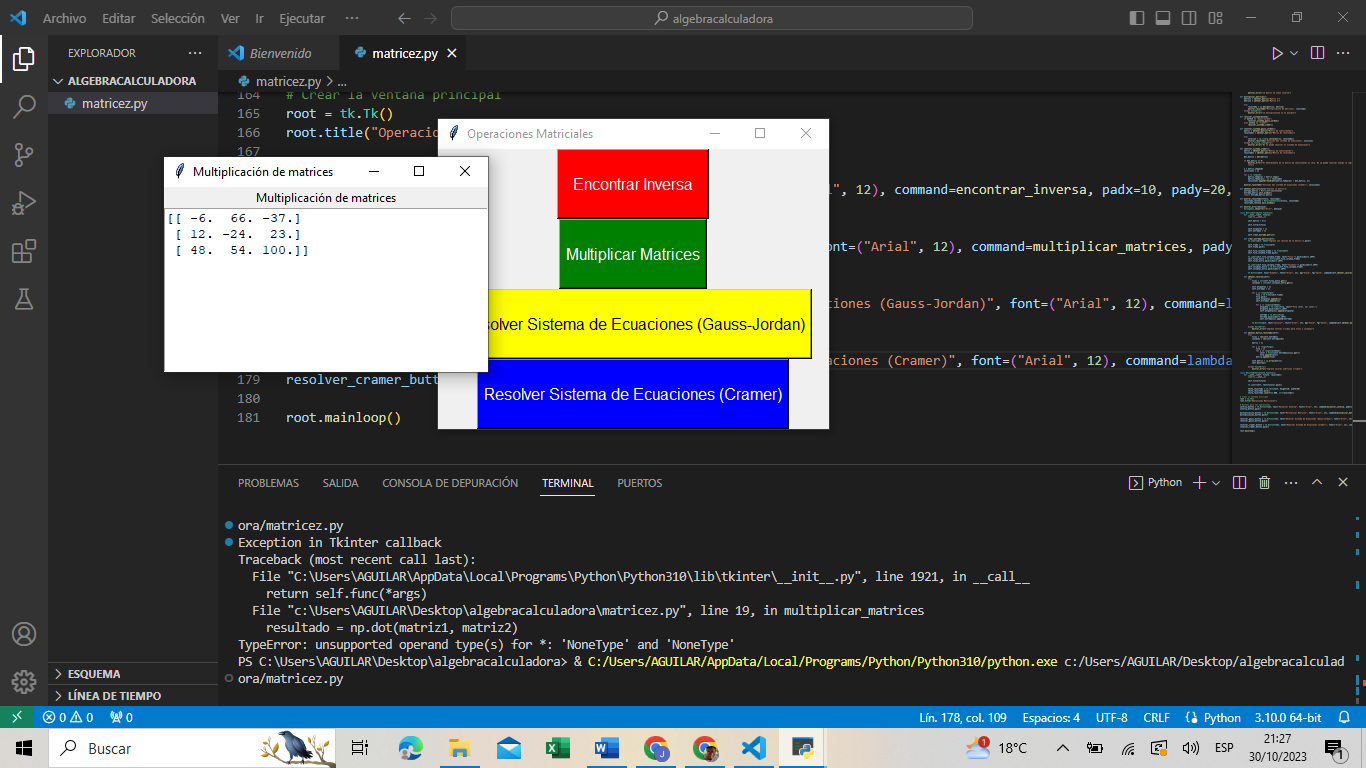
por



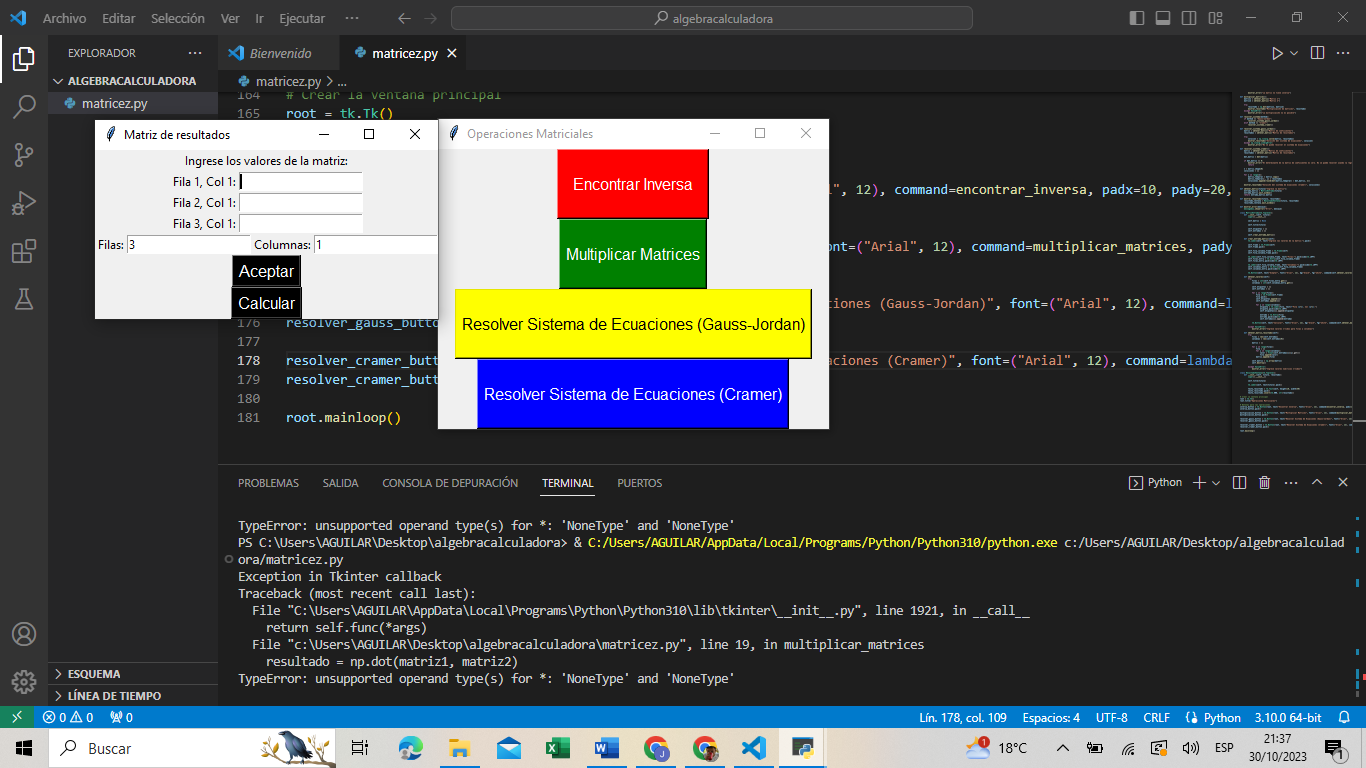


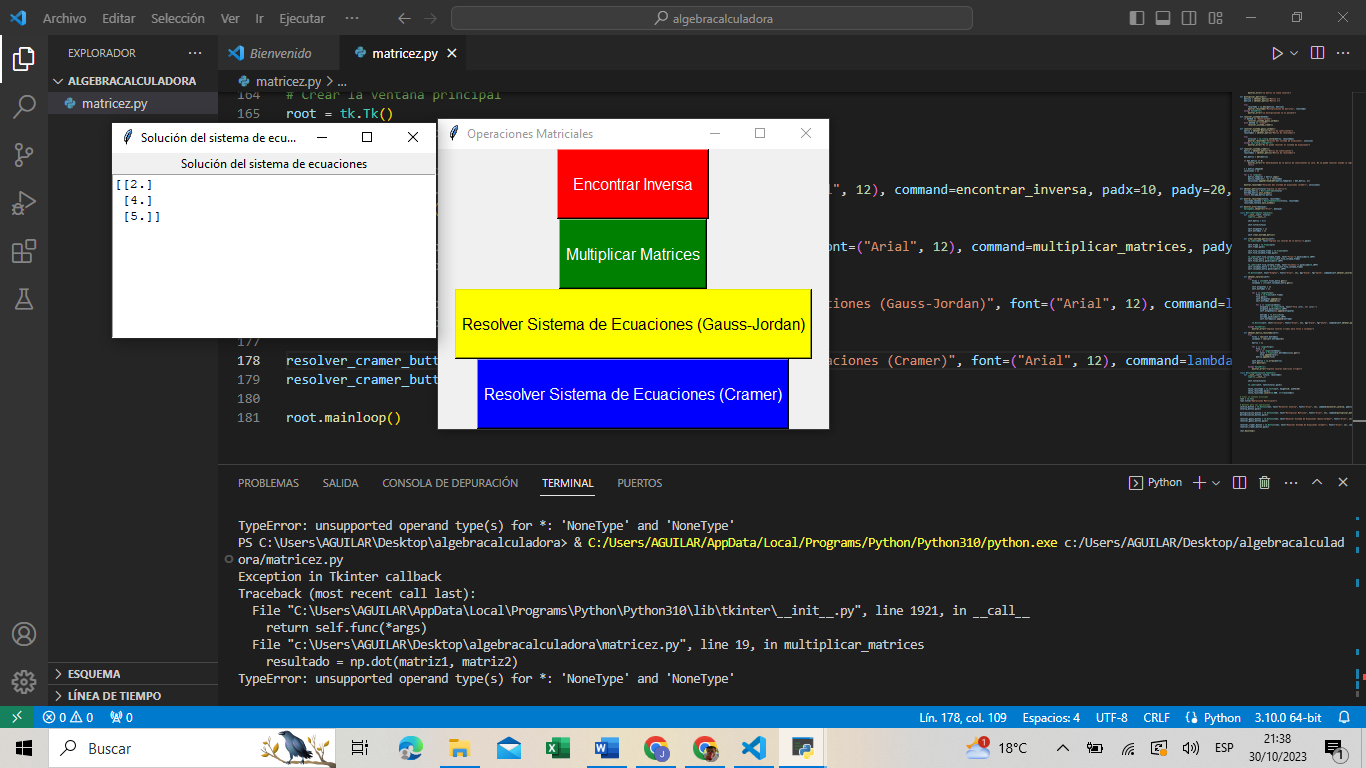
MULTIPLICACION DE MATRICES 3x3

por

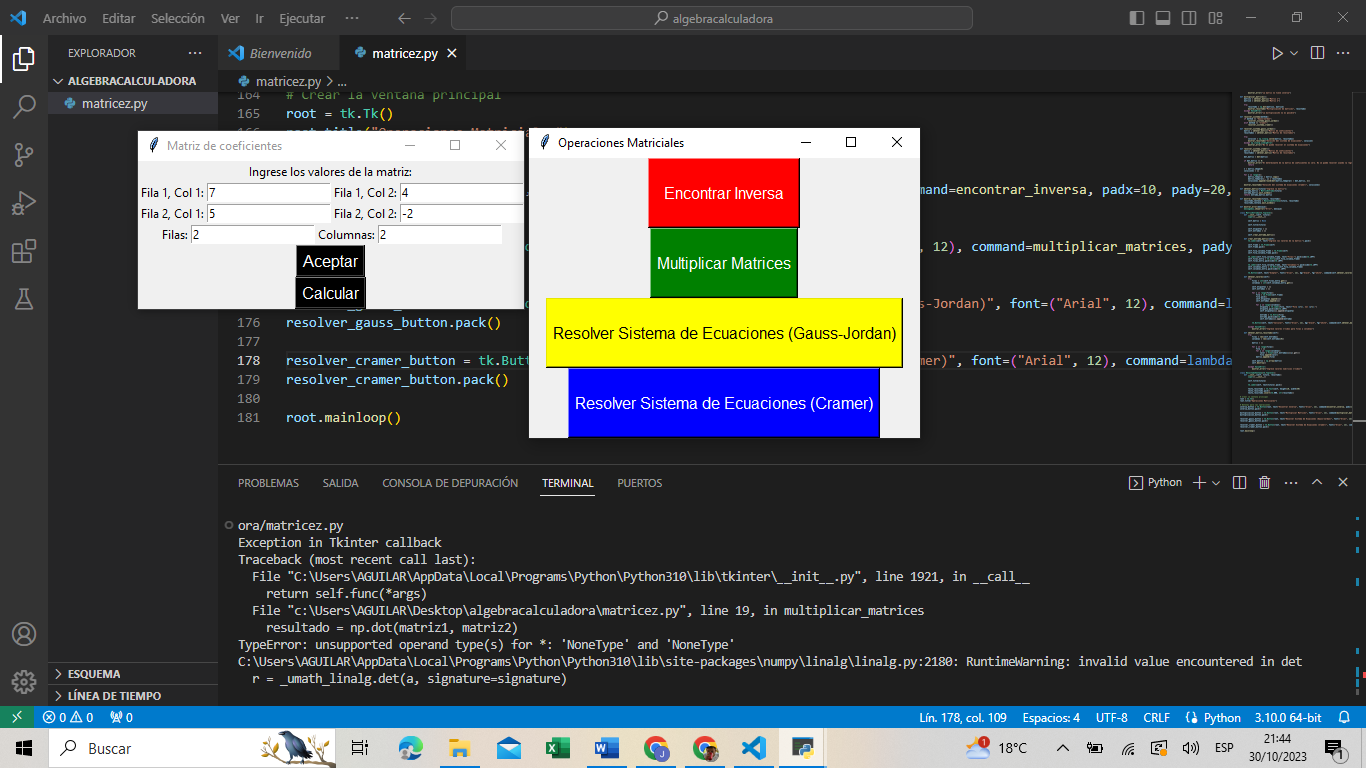
 

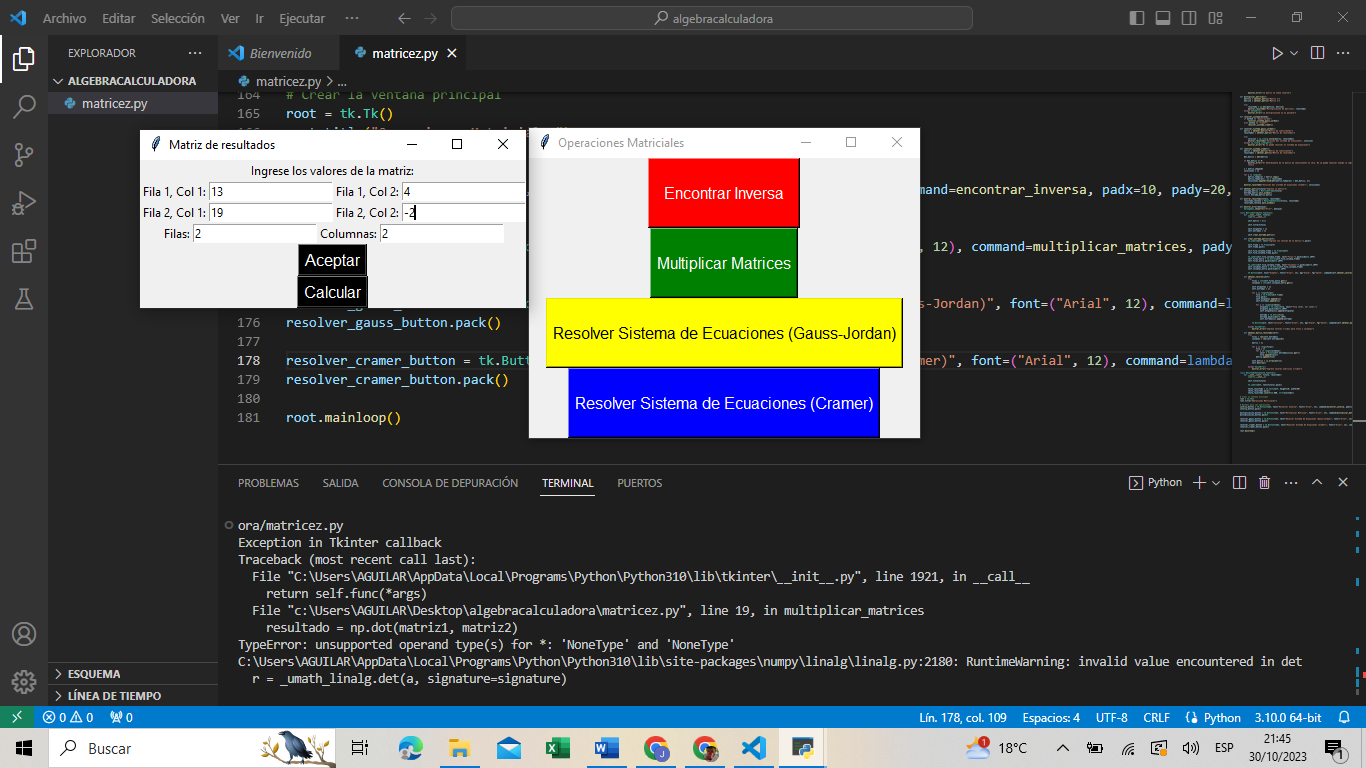
Solución de Sistemas de Ecuaciones (Gauss Jordan)





Solución de Sistemas de Ecuaciones (Cramer)





CODIGO APLICACIÓN

import tkinter as tk

import numpy as np

from numpy.linalg import inv, det

from tkinter import messagebox

def encontrar\_inversa():

    matriz = obtener\_matriz()

    try:

        inversa = inv(matriz)

        mostrar\_resultado("Inversa de la matriz", inversa)

    except np.linalg.LinAlgError:

        mostrar\_error("La matriz no tiene inversa")

def multiplicar\_matrices():

    matriz1 = obtener\_matriz("Matriz 1")

    matriz2 = obtener\_matriz("Matriz 2")

    try:

        resultado = np.dot(matriz1, matriz2)

        mostrar\_resultado("Multiplicación de matrices", resultado)

    except ValueError:

        mostrar\_error("La multiplicación no es posible")

def resolver\_sistema(method):

    if method == "Gauss-Jordan":

        resolver\_sistema\_gauss\_jordan()

    elif method == "Cramer":

        resolver\_sistema\_cramer()

def resolver\_sistema\_gauss\_jordan():

    matriz = obtener\_matriz("Matriz de coeficientes")

    resultados = obtener\_matriz("Matriz de resultados")

    try:

        solucion = np.linalg.solve(matriz, resultados)

        mostrar\_resultado("Solución del sistema de ecuaciones", solucion)

    except np.linalg.LinAlgError:

        mostrar\_error("No se puede resolver el sistema de ecuaciones")

def resolver\_sistema\_cramer():

    matriz = obtener\_matriz("Matriz de coeficientes")

    resultados = obtener\_matriz("Matriz de resultados")

    det\_matriz = det(matriz)

    if det\_matriz == 0:

        mostrar\_error("El determinante de la matriz de coeficientes es cero. No se puede resolver usando la regla de Cramer.")

        return

    n = matriz.shape[0]

    soluciones = []

    for i in range(n):

        matriz\_temporal = matriz.copy()

        matriz\_temporal[:, i] = resultados

        soluciones.append(round(det(matriz\_temporal) / det\_matriz, 2))

    mostrar\_resultado("Solución del sistema de ecuaciones (Cramer)", soluciones)

def obtener\_matriz(titulo="Ingrese la matriz"):

    entrada\_matriz = MatrizEntrada(titulo)

    entrada\_matriz.wait\_window()

    return entrada\_matriz.matriz

def mostrar\_resultado(titulo, resultado):

    resultado\_ventana = ResultadoVentana(titulo, resultado)

    resultado\_ventana.wait\_window()

def mostrar\_error(mensaje):

    messagebox.showerror("Error", mensaje)

class MatrizEntrada(tk.Toplevel):

    def \_\_init\_\_(self, titulo):

        super().\_\_init\_\_()

        self.matriz = None

        self.title(titulo)

        self.etiquetas = []

        self.entradas = []

        self.crear\_entrada\_matriz()

    def crear\_entrada\_matriz(self):

        tk.Label(self, text="Ingrese los valores de la matriz:").pack()

        self.frame = tk.Frame(self)

        self.frame.pack()

        self.fila\_columna\_frame = tk.Frame(self)

        self.fila\_columna\_frame.pack()

        tk.Label(self.fila\_columna\_frame, text="Filas:").pack(side=tk.LEFT)

        self.filas\_entry = tk.Entry(self.fila\_columna\_frame)

        self.filas\_entry.pack(side=tk.LEFT)

        tk.Label(self.fila\_columna\_frame, text="Columnas:").pack(side=tk.LEFT)

        self.columnas\_entry = tk.Entry(self.fila\_columna\_frame)

        self.columnas\_entry.pack(side=tk.LEFT)

        tk.Button(self, text="Aceptar", font=("Arial", 12), bg="black", fg="white", command=self.obtener\_valores).pack()

    def obtener\_valores(self):

        try:

            filas = int(self.filas\_entry.get())

            columnas = int(self.columnas\_entry.get())

            self.etiquetas = []

            self.entradas = []

            for i in range(filas):

                fila = tk.Frame(self.frame)

                fila.pack()

                self.etiquetas.append([])

                self.entradas.append([])

                for j in range(columnas):

                    etiqueta = tk.Label(fila, text=f"Fila {i+1}, Col {j+1}:")

                    etiqueta.pack(side=tk.LEFT)

                    self.etiquetas[i].append(etiqueta)

                    entrada = tk.Entry(fila)

                    entrada.pack(side=tk.LEFT)

                    self.entradas[i].append(entrada)

            tk.Button(self, text="Calcular", font=("Arial", 12), bg="black", fg="white", command=self.obtener\_matriz\_resultado).pack()

        except ValueError:

            mostrar\_error("Ingrese valores válidos para filas y columnas")

    def obtener\_matriz\_resultado(self):

        try:

            filas = len(self.entradas)

            columnas = len(self.entradas[0])

            matriz = []

            for i in range(filas):

                fila = []

                for j in range(columnas):

                    valor = float(self.entradas[i][j].get())

                    fila.append(valor)

                matriz.append(fila)

            self.matriz = np.array(matriz)

            self.destroy()

        except ValueError:

            mostrar\_error("Ingrese valores numéricos válidos")

class ResultadoVentana(tk.Toplevel):

    def \_\_init\_\_(self, titulo, resultado):

        super().\_\_init\_\_()

        self.title(titulo)

        tk.Label(self, text=titulo).pack()

        texto\_resultado = tk.Text(self, height=10, width=40)

        texto\_resultado.pack()

        texto\_resultado.insert(tk.END, str(resultado))

# Crear la ventana principal

root = tk.Tk()

root.title("Operaciones Matriciales")

# Botones para las operaciones

inversa\_button = tk.Button(root, text="Encontrar Inversa", font=("Arial", 12), command=encontrar\_inversa, padx=10, pady=20, bg="red", fg="white")

inversa\_button.pack()

multiplicacion\_button = tk.Button(root, text="Multiplicar Matrices", font=("Arial", 12), command=multiplicar\_matrices, pady=20, bg="green", fg="white")

multiplicacion\_button.pack()

resolver\_gauss\_button = tk.Button(root, text="Resolver Sistema de Ecuaciones (Gauss-Jordan)", font=("Arial", 12), command=lambda: resolver\_sistema("Gauss-Jordan"), pady=20, bg="yellow")

resolver\_gauss\_button.pack()

resolver\_cramer\_button = tk.Button(root, text="Resolver Sistema de Ecuaciones (Cramer)", font=("Arial", 12), command=lambda: resolver\_sistema("Cramer"), pady=20, bg="blue", fg="white")

resolver\_cramer\_button.pack()

root.mainloop()