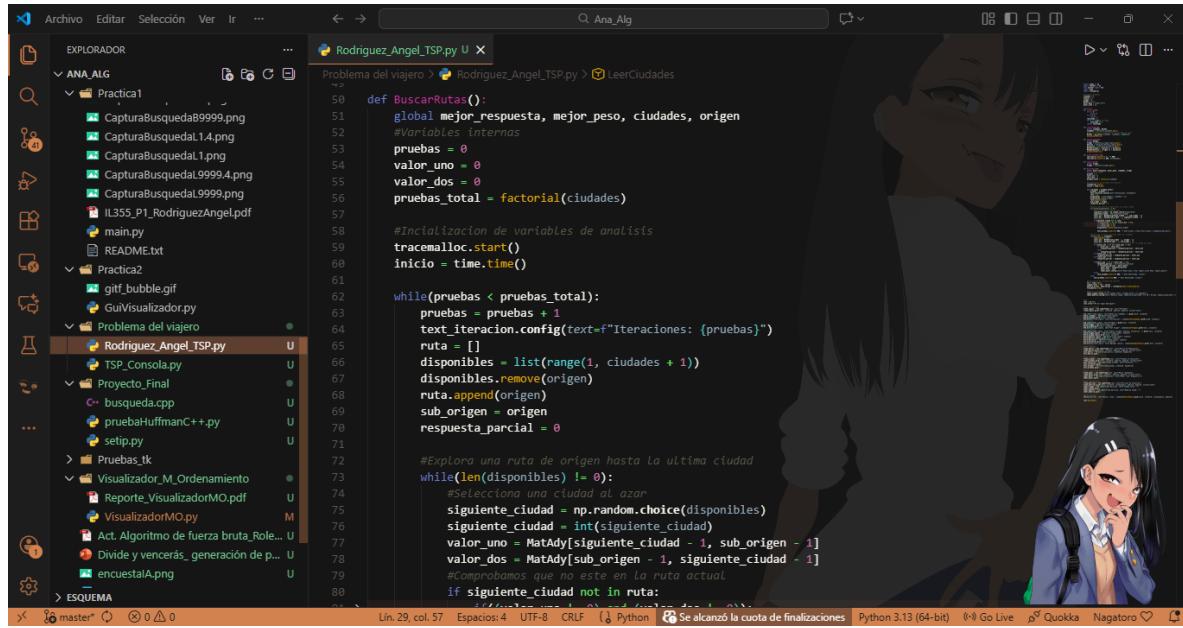


## Parte del código de búsqueda



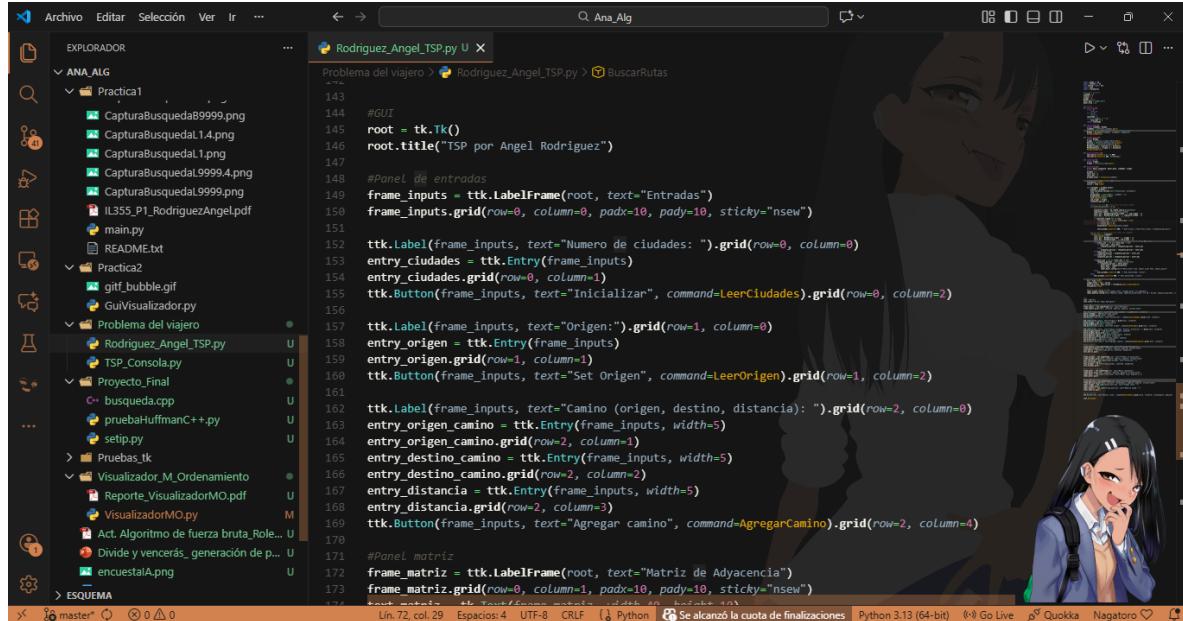
```
def BuscarRutas():
    global mejor_resposta, mejor_peso, ciudades, origen
    #Variables internas
    pruebas = 0
    valor_uno = 0
    valor_dos = 0
    pruebas_total = factorial(ciudades)

    #Inicializacion de variables de analisis
    tracemalloc.start()
    inicio = time.time()

    while(pruebas < pruebas_total):
        pruebas = pruebas + 1
        text_iteracion.config(text=f"Iteraciones: {pruebas}")
        ruta = []
        disponibles = list(range(1, ciudades + 1))
        disponibles.remove(origen)
        ruta.append(origen)
        sub_origen = origen
        respuesta_parcial = 0

        #Explora una ruta de origen hasta la ultima ciudad
        while(len(disponibles) != 0):
            #selecciona una ciudad al azar
            siguiente_ciudad = np.random.choice(disponibles)
            siguiente_ciudad = int(siguiente_ciudad)
            valor_uno = MatAdy[siguiente_ciudad - 1, sub_origen - 1]
            valor_dos = MatAdy[sub_origen - 1, siguiente_ciudad - 1]
            #comprobamos que no este en la ruta actual
            if siguiente_ciudad not in ruta:
```

## Parte del código de GUI



```
#GUI
root = tk.Tk()
root.title("TSP por Angel Rodriguez")

#Panel de entradas
frame_inputs = ttk.LabelFrame(root, text="Entradas")
frame_inputs.grid(row=0, column=0, padx=10, pady=10, sticky="nsew")

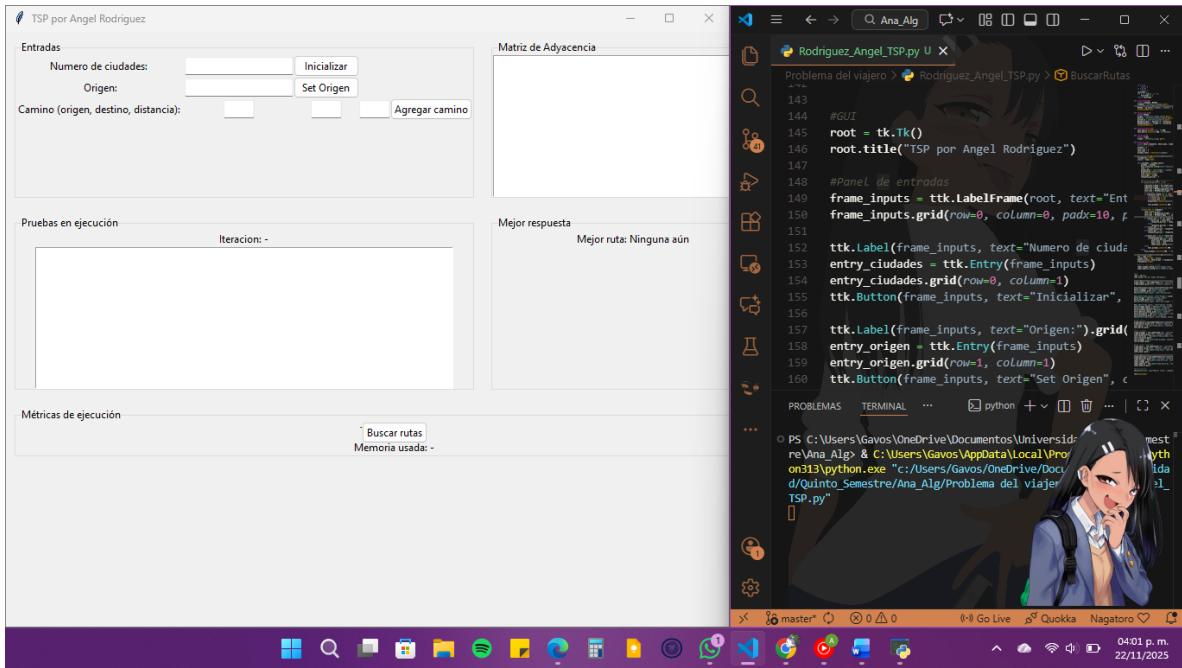
ttk.Label(frame_inputs, text="Número de ciudades:").grid(row=0, column=0)
entry_ciudades = ttk.Entry(frame_inputs)
entry_ciudades.grid(row=0, column=1)
ttk.Button(frame_inputs, text="Inicializar", command=LeerCiudades).grid(row=0, column=2)

ttk.Label(frame_inputs, text="Origen:").grid(row=1, column=0)
entry_origen = ttk.Entry(frame_inputs)
entry_origen.grid(row=1, column=1)
ttk.Button(frame_inputs, text="Set Origen", command=LeerOrigen).grid(row=1, column=2)

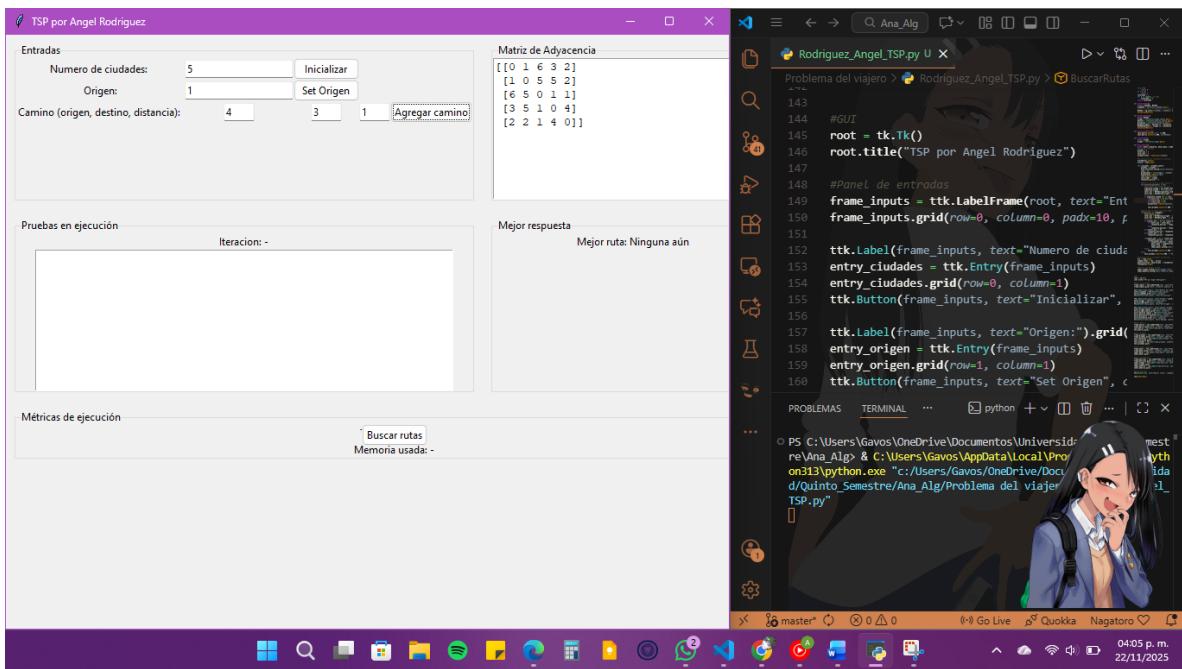
ttk.Label(frame_inputs, text="Camino (origen, destino, distancia):").grid(row=2, column=0)
entry_origen_camino = ttk.Entry(frame_inputs, width=5)
entry_origen_camino.grid(row=2, column=1)
entry_destino_camino = ttk.Entry(frame_inputs, width=5)
entry_destino_camino.grid(row=2, column=2)
entry_distancia = ttk.Entry(frame_inputs, width=5)
entry_distancia.grid(row=2, column=3)
ttk.Button(frame_inputs, text="Agregar camino", command=AgregarCamino).grid(row=2, column=4)

#Panel matriz
frame_matriz = ttk.LabelFrame(root, text="Matriz de Adyacencia")
frame_matriz.grid(row=0, column=1, padx=10, pady=10, sticky="nsew")
```

## Ejecución del programa



### Inserción completa de las ciudades (nodos)



Resultado de la búsqueda con origen en el primer nodo

**TSP por Angel Rodriguez**

**Entradas**

Número de ciudades:  Inicializar  
 Origen:  Set Origen  
 Camino (origen, destino, distancia):    Agregar camino

**Pruebas en ejecución**  
 Iteraciones: 120

```
Ruta actual: [1, 5] Peso actual: 2 Ruta actual: [1, 5, 2] Peso actual: 4 Ruta actual: [1, 5, 2, 4] Peso actual: 9 Ruta actual: [1, 5, 2, 4, 3] Peso actual: 10 Ruta actual: [1, 5] Peso actual: 2 Ruta actual: [1, 5, 4] Peso actual: 6 Ruta actual: [1, 5, 4, 2] Peso actual: 11 Ruta actual: [1, 5, 4, 2, 3] Peso actual: 16 Ruta actual: [1, 4] Peso actual: 3 Ruta actual: [1, 4, 2] Peso actual: 8 Ruta actual: [1, 4, 2, 5] Peso actual: 10 Ruta actual: [1, 4, 2, 5, 3] Peso actual: 11 Ruta actual: [1, 2] Peso actual: 4 Ruta actual: [1, 2, 4] Peso actual: 6 Ruta actual: [1, 2, 4, 3] Peso actual: 7 Ruta a
```

**Métricas de ejecución**

Tiempo: Buscar rutas: 5 segundos  
 Memoria usada: 986.43 KB | Máxima: 987.81 KB

**Matriz de Adyacencia**

```
[[0 1 6 3 2]
 [1 0 5 5 2]
 [6 5 0 1 1]
 [3 5 1 0 4]
 [2 2 1 4 0]]
```

**Mejor respuesta**  
 Nueva mejor ruta: [1, 2, 5, 3, 4, 1] Peso: 8

**Rodriguez\_Angel\_TSP.py**

```
Problema del viajero > Rodriguez_Angel_TSP.py > BuscarRutas
143
144 #GUI
145 root = tk.Tk()
146 root.title("TSP por Angel Rodriguez")
147
148 panel_de_entradas
149 frame_inputs = ttk.LabelFrame(root, te
150 frame_inputs.grid(row=0, column=0, pad
151
152 ttk.Label(frame_inputs, text="Número d
153 entry_ciudades = ttk.Entry(frame_inputs
154 entry_ciudades.grid(row=0, column=1)
155 ttk.Button(frame_inputs, text="Inicial
156
157 ttk.Label(frame_inputs, text="Origen:")
158 entry_origen = ttk.Entry(frame_inputs)
159 entry_origen.grid(row=1, column=1)
160 ttk.Button(frame_inputs, text="Set Ori
```

**TERMINAL**

```
PS C:\Users\Gavos\OneDrive\Documentos\Universidad\Quinto Semestre\Algo & C:\Users\Gavos\Algoritmos\Python3\python.exe "c:/Users/Gavos/OneDrive/Documentos/Universidad/Quinto Semestre/Algoritmos/Python3/Rodriguez_Angel_TSP.py"
04:07 p. m.
22/11/2025
```

<img alt="Screenshot of a Windows desktop showing a Python application window for a Traveling Salesman Problem (TSP) solver. The application has tabs for 'Entradas' (Inputs), 'Matriz de Adyacencia' (Adjacency Matrix), 'Pruebas en ejecución' (Execution Tests), and 'Métricas de ejecución' (Execution Metrics). The 'Pruebas en ejecución' tab shows a log of route iterations and weights. The 'Matriz de Adyacencia' tab displays an adjacency matrix for 5 cities. The 'Métricas de ejecución' tab shows execution time and memory usage. To the right of the application is a code editor window showing the Python script 'Rodriguez\_Angel\_TSP.py'. Below the application windows is a taskbar with various icons and system status indicators like battery level and network connection.</pre>