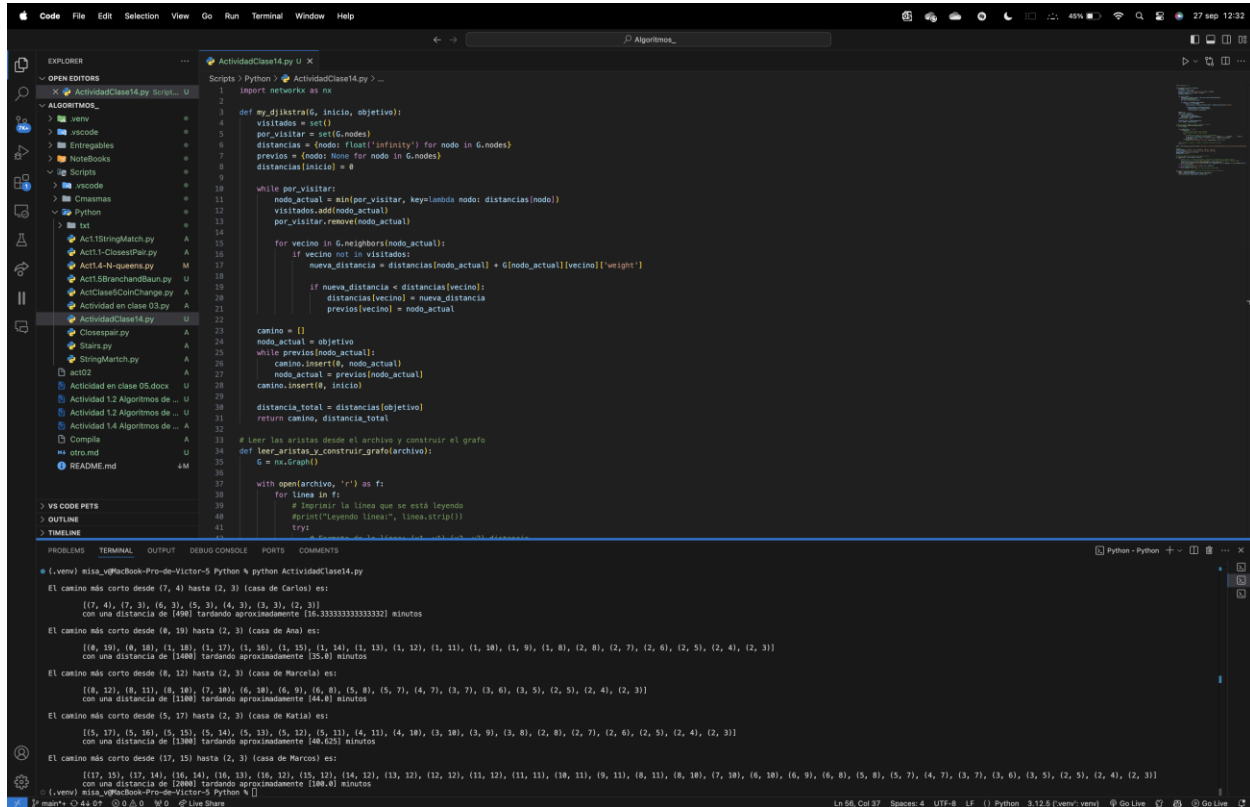


Actividad de Clase 14 - Reunion en casa de Luis

Captura de pantalla del codigo funcionando



```
1 import networkx as nx
2
3 def my_dijkstra(G, inicio, objetivo):
4     visitados = set()
5     per_visitar = set(G.nodes)
6     distancias = {nodo: float('infinity') for nodo in G.nodes}
7     previos = {nodo: None for nodo in G.nodes}
8     distancias[inicio] = 0
9
10    while per_visitar:
11        nodo_actual = min(per_visitar, key=lambda nodo: distancias[nodo])
12        visitados.add(nodo_actual)
13        per_visitar.remove(nodo_actual)
14
15        for vecino in G.neighbors(nodo_actual):
16            if vecino not in visitados:
17                nueva_distancia = distancias[nodo_actual] + G[nodo_actual][vecino]['weight']
18
19                if nueva_distancia < distancias[vecino]:
20                    distancias[vecino] = nueva_distancia
21                    previos[vecino] = nodo_actual
22
23    camino = []
24    nodo_actual = objetivo
25    while previos[nodo_actual]:
26        camino.insert(0, nodo_actual)
27        nodo_actual = previos[nodo_actual]
28    camino.insert(0, inicio)
29
30    distancia_total = distancias[objetivo]
31    return camino, distancia_total
32
33 # Leer las aristas desde el archivo y construir el grafo
34 def leer_aristas_y_construir_grafo(archivo):
35     G = nx.Graph()
36
37     with open(archivo, "r") as f:
38         for linea in f:
39             # Imprimir la linea que se está leyendo
40             #print("Leyendo linea:", linea.strip())
41             try:
42                 # Parsear la linea para obtener origen, destino y peso
43                 origen, destino, peso = linea.strip().split(',')
44                 G.add_edge(origen, destino, weight=int(peso))
45
46     return G
47
48 # Ejemplo de uso
49 if __name__ == '__main__':
50     G = leer_aristas_y_construir_grafo('grafo.txt')
51     inicio = 'A'
52     objetivo = 'F'
53     camino, distancia = my_dijkstra(G, inicio, objetivo)
54     print("Camino más corto de", inicio, "a", objetivo, "es:", camino)
55     print("Distancia total:", distancia)
```

PROBLEMS TERMINAL OUTPUT DEBUG CONSOLE PORTS COMMENTS

```
Python 3.12.5 [venv: venv]
(.venv) miss_v@MacBook-Pro-de-Victor-5: Python 4 python ActividadClase14.py
El camino más corto desde (7, 4) hasta (2, 3) (casa de Carlos) es:
[(7, 4), (7, 3), (6, 3), (5, 3), (4, 3), (3, 3), (2, 3)]
con una distancia de [498] tardando aproximadamente [16.333333333333332] minutos

El camino más corto desde (8, 19) hasta (2, 3) (casa de Ana) es:
[(8, 19), (8, 18), (1, 17), (1, 16), (1, 15), (1, 14), (1, 13), (1, 12), (1, 11), (1, 10), (1, 9), (1, 8), (2, 8), (2, 7), (2, 6), (2, 5), (2, 4), (2, 3)]
con una distancia de [1488] tardando aproximadamente [35.6] minutos

El camino más corto desde (8, 12) hasta (2, 3) (casa de Marcela) es:
[(8, 12), (8, 11), (5, 10), (7, 10), (6, 10), (6, 9), (6, 8), (5, 8), (5, 7), (4, 7), (3, 7), (3, 6), (3, 5), (2, 5), (2, 4), (2, 3)]
con una distancia de [1188] tardando aproximadamente [64.8] minutos

El camino más corto desde (5, 17) hasta (2, 3) (casa de Katia) es:
[(5, 17), (5, 16), (5, 15), (5, 14), (5, 13), (5, 12), (5, 11), (4, 11), (4, 10), (3, 10), (3, 9), (3, 8), (2, 8), (2, 7), (2, 6), (2, 5), (2, 4), (2, 3)]
con una distancia de [1388] tardando aproximadamente [68.625] minutos

El camino más corto desde (17, 15) hasta (2, 3) (casa de Marcos) es:
[(17, 15), (17, 14), (16, 14), (16, 13), (16, 12), (15, 12), (13, 12), (12, 12), (11, 12), (11, 11), (10, 11), (9, 11), (8, 11), (8, 10), (7, 10), (6, 10), (6, 9), (6, 8), (5, 8), (5, 7), (4, 7), (3, 7), (3, 6), (3, 5), (2, 5), (2, 4), (2, 3)]
con una distancia de [2688] tardando aproximadamente [189.8] minutos

(.venv) miss_v@MacBook-Pro-de-Victor-5: Python 4
```

Enlace a repositorio

[\[https://prod.liveshare.vsengsaas.visualstudio.com/join?D25DA6CB582112367871B8F099653F7CD654\]](https://prod.liveshare.vsengsaas.visualstudio.com/join?D25DA6CB582112367871B8F099653F7CD654)