**DFS -Depth First Search**

**Code**

import sqlite3

**# Create a database connection and cursor**

connection = sqlite3.connect("visited\_places.db")

cursor = connection.cursor()

**# Create a table to store visited places**

cursor.execute('''CREATE TABLE IF NOT EXISTS visited\_places

(vertex TEXT PRIMARY KEY, visit\_order INTEGER)''')

connection.commit()

def dfs(graph, start, end, visited=None, path=None):

if visited is None:

visited = set()

if path is None:

path = []

visited.add(start)

path.append(vertices[start]) # Append the actual place name

**# Store visited place in the database**

vertex\_name = vertices[start]

visit\_order = len(path)

**# Check if the vertex already exists in the table**

cursor.execute("SELECT \* FROM visited\_places WHERE vertex = ?", (vertex\_name,))

existing\_row = cursor.fetchone()

if existing\_row is None:

# If the vertex doesn't exist, insert it

cursor.execute("INSERT INTO visited\_places (vertex, visit\_order) VALUES (?, ?)",

(vertex\_name, visit\_order))

else:

# If the vertex already exists, update the visit\_order

cursor.execute("UPDATE visited\_places SET visit\_order = ? WHERE vertex = ?",

(visit\_order, vertex\_name))

connection.commit()

if start == end:

return path

for neighbor in range(len(graph[start])):

if graph[start][neighbor] == 1 and neighbor not in visited:

new\_path = dfs(graph, neighbor, end, visited, path)

if new\_path:

return new\_path

path.pop()

visited.remove(start)

return None

graph = [

[0,1,1,0,0,0,0,0],

[1,0,1,1,1,0,0,0],

[1,1,0,1,0,0,1,0],

[0,1,1,0,1,0,0,1],

[0,1,0,1,0,0,0,1],

[0,0,0,0,0,0,0,1],

[0,0,1,0,0,0,0,0],

[0,0,0,1,1,1,0,0]

]

vertices = ['Devgad','Malvan','Kankavli', 'Kudal','Vengurla','Dodamargh','Vaibhavwadi','Sawantwadi']

start\_vertex = int(input("Enter your Start index value: "))

end\_vertex =int(input("Enter your destination index value: "))

result = dfs(graph, start\_vertex, end\_vertex)

if result:

print(f"Path from {vertices[start\_vertex]} to {vertices[end\_vertex]}:")

print(' -> '.join(result)) # Join the place names with ' -> ' separator

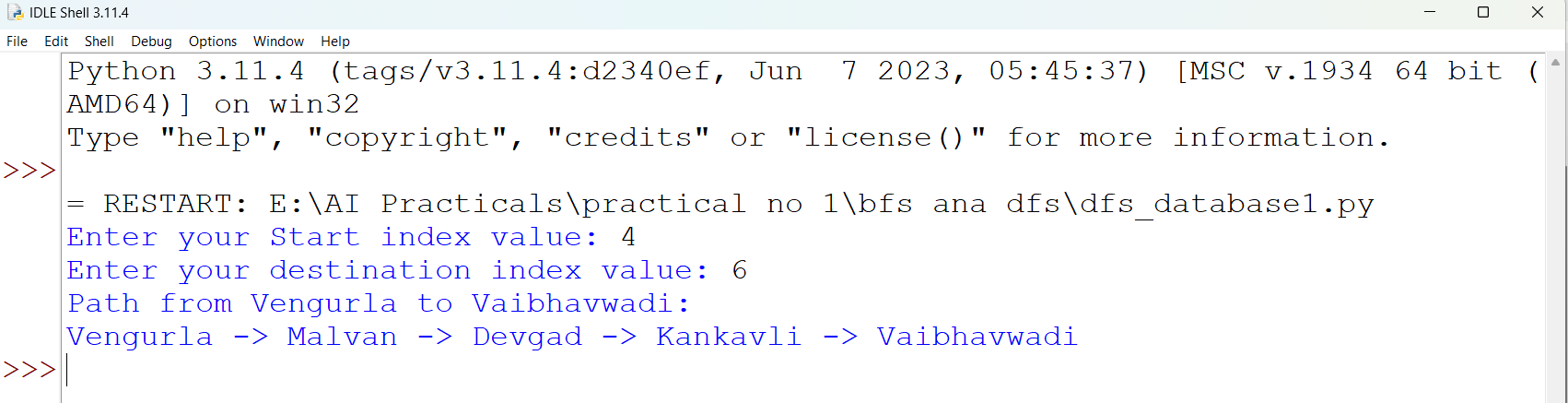
else:

print(f"No path found from {vertices[start\_vertex]} to {vertices[end\_vertex]}")

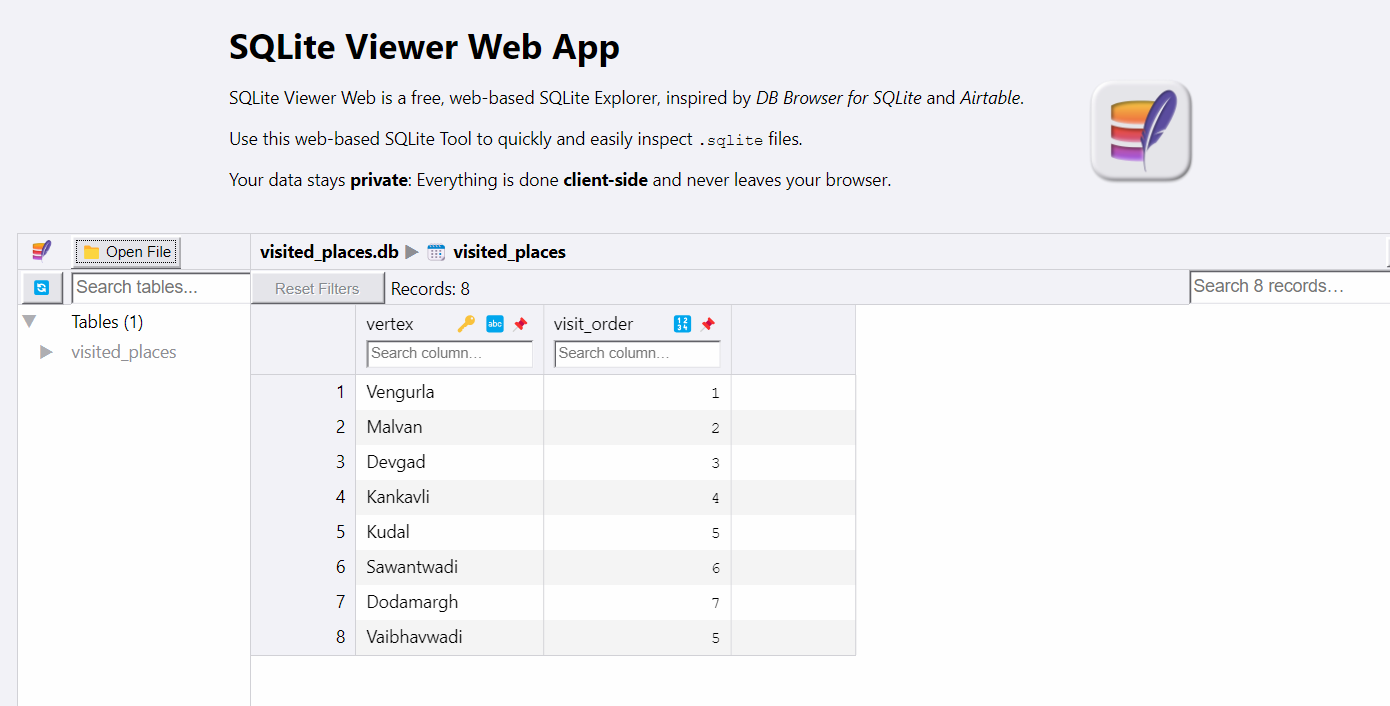
**# Close the database connection**

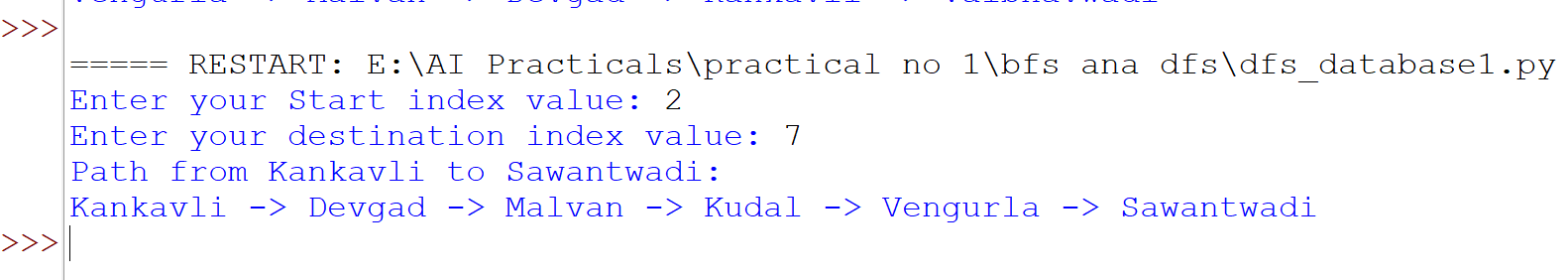
connection.close()

**OUTPUT**

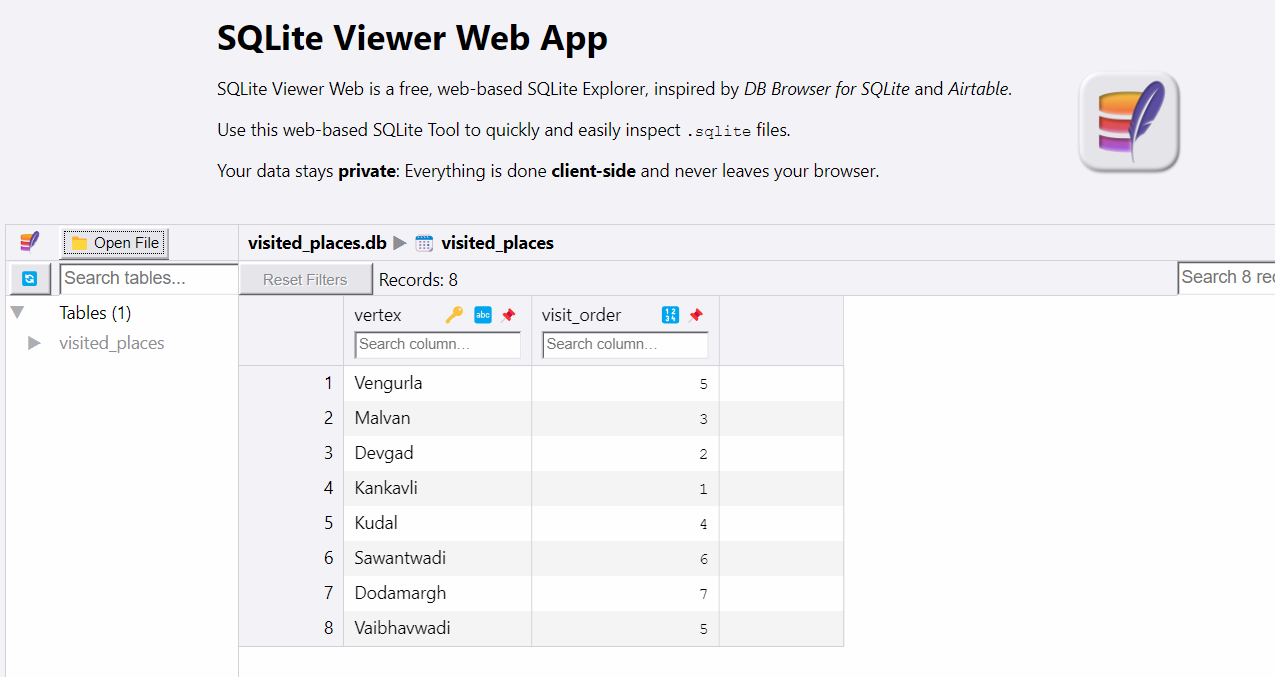
****

**In SQLite database all Visited Place are stored**

****

****

**Database**

****

**BFS -Breadth First Search**

**Code**

from collections import deque

def bfs(graph, start, end):

visited = set()

queue = deque([[start]])

while queue:

path = queue.popleft()

vertex = path[-1]

if vertex == end:

return path

if vertex not in visited:

visited.add(vertex)

for neighbor in range(len(graph[vertex])):

if graph[vertex][neighbor] == 1:

new\_path = list(path)

new\_path.append(neighbor)

queue.append(new\_path)

return None

**# Define the graph as an adjacency matrix**

graph = [

[0,1,1,0,0,0,0,0],

[1,0,1,1,1,0,0,0],

[1,1,0,1,0,0,1,0],

[0,1,1,0,1,0,0,1],

[0,1,0,1,0,0,0,1],

[0,0,0,0,0,0,0,1],

[0,0,1,0,0,0,0,0],

[0,0,0,1,1,1,0,0]

]

vertices = ['devgad','malvan','kankavli', 'kudal','vengurla','dodamatg','vaibhavwadi','sawantwadi']

start\_vertex =int(input("Enter Your Start location index value: "))

end\_vertex = int(input("Enter Your destination location index value: "))

result = bfs(graph, start\_vertex, end\_vertex)

if result:

print(f"Path from {vertices[start\_vertex]} to {vertices[end\_vertex]}:")

for i, vertex in enumerate(result):

print(vertices[vertex], end='')

if i < len(result) - 1:

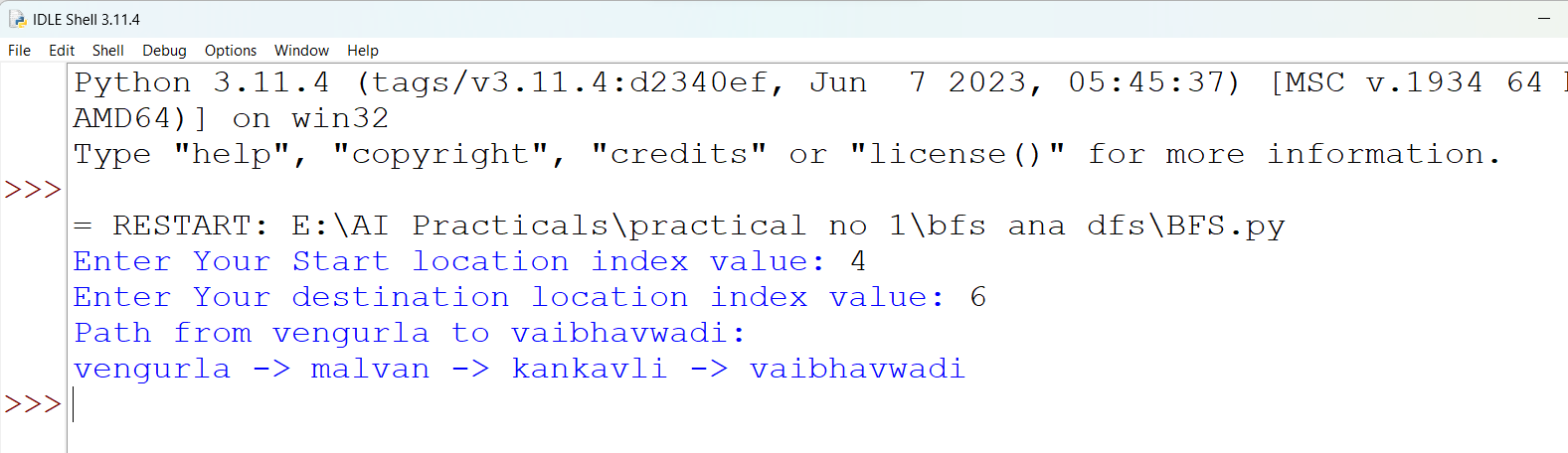
print(' -> ', end='')

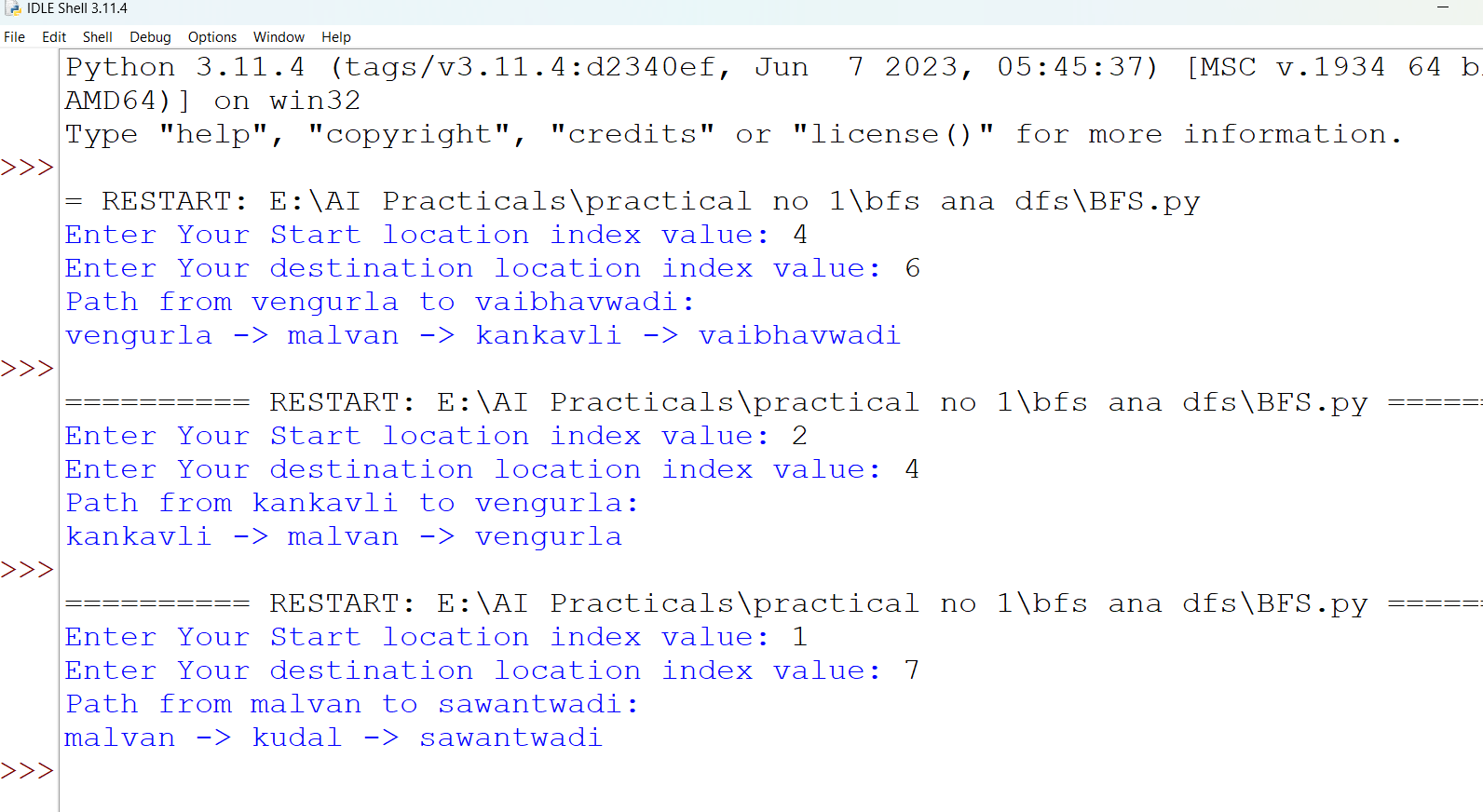
print()

else:

print(f"No path found from {vertices[start\_vertex]} to {vertices[end\_vertex]}")

**OUTPUT**

****

****