**LAB 06**

**Implement the discussion of TSP in python.**

import itertools

# Given graph with 4 edges

graph = {

    1: {2: 10, 3: 15, 4: 20},

    2: {1: 10, 3: 35, 4: 25},

    3: {1: 15, 2: 35, 4: 30},

    4: {1: 20, 2: 25, 3: 30},

}

start\_edge = 1

def TSP(graph, start\_edge):

    # Generate all permutations of cities

    cities = list(graph.keys())

    cities.remove(start\_edge)

    permutations = list(itertools.permutations(cities))

    # Add start\_edge to each permutation and calculate the total distance

    min\_distance = float("inf")

    optimal\_route = None

    for permutation in permutations:

        distance = graph[start\_edge][permutation[0]]

        for i in range(len(permutation) - 1):

            distance += graph[permutation[i]][permutation[i + 1]]

        distance += graph[permutation[-1]][start\_edge]

        # Update the minimum distance and optimal route

        if distance < min\_distance:

            min\_distance = distance

            optimal\_route = (start\_edge,) + permutation + (start\_edge,)

    return optimal\_route, min\_distance

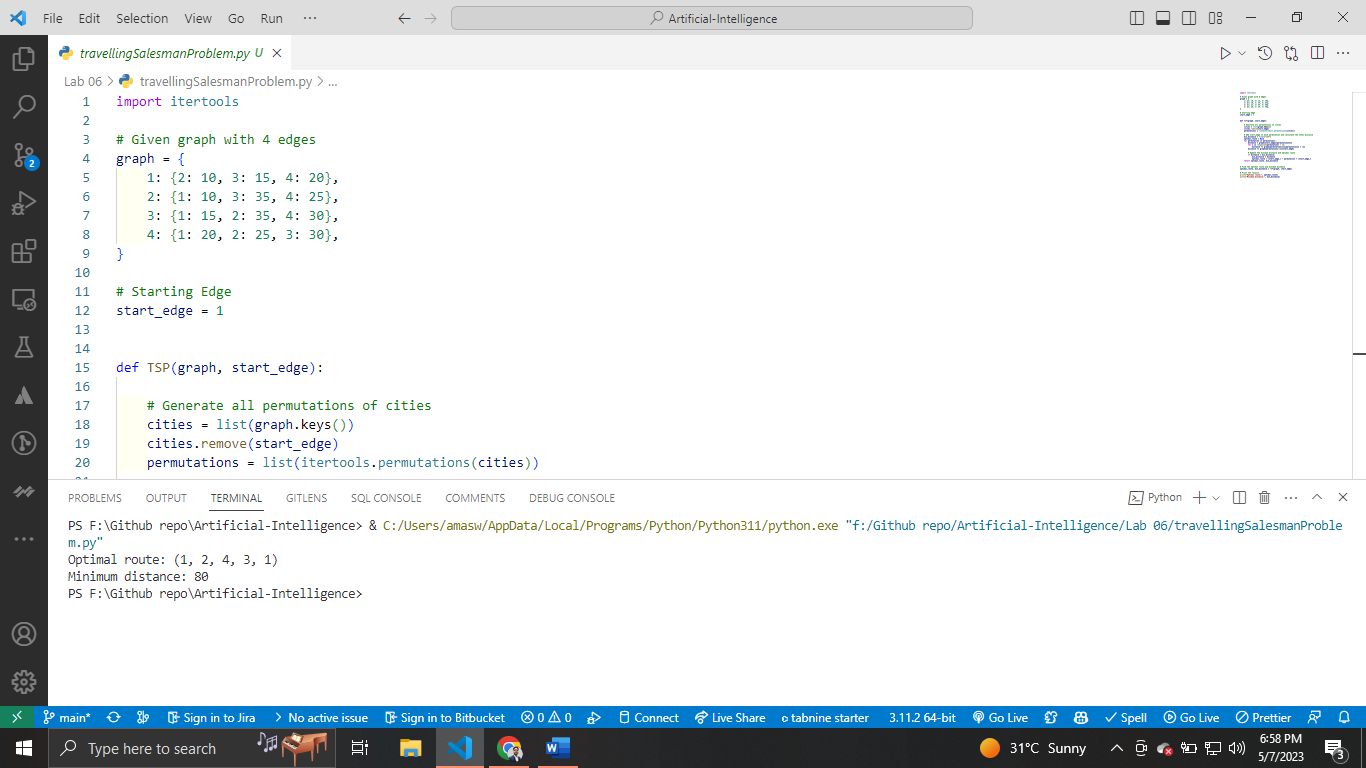
# Find the optimal route and minimum distance

optimal\_route, min\_distance = TSP(graph, start\_edge)

# Print the results

print("Optimal route:", optimal\_route)

print("Minimum distance:", min\_distance)



**Implement the discussed approach for tower of Hanoi in Python Language.**

def tower\_of\_hanoi(n, source, destination, temporary, step\_count):

    """

    Parameters:

    - n: The number of disks to move.

    - source: The peg from which to move the disks.

    - destination: The peg to which to move the disks.

    - temporary: The peg to use as a temporary holding area.

    - step\_count: The number of steps taken so far.

    Returns:

    - The number of steps taken to solve the problem.

    """

    if n == 1:

        step\_count += 1

        print(f"{step\_count}. Move disk 1 from {source} to {destination}")

        return step\_count

    else:

        step\_count = tower\_of\_hanoi(

            n - 1, source, temporary, destination, step\_count)

        step\_count += 1

        print(f"{step\_count}. Move disk {n} from {source} to  {destination}")

        step\_count = tower\_of\_hanoi(

            n - 1, temporary, destination, source, step\_count)

        return step\_count

step\_count = 0

step\_count = tower\_of\_hanoi(3, "A", "C", "B", step\_count)

print(f"Total number of steps: {step\_count}")

