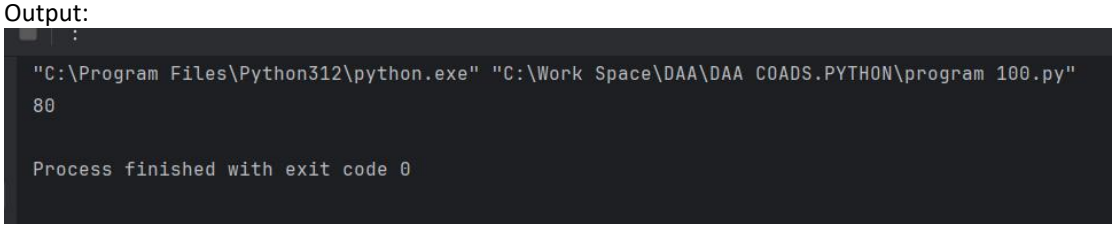
Program 100.Dynamic Programming

Program:

```
def tsp(graph):
    n = len(graph)  # Number of cities

    # dp[mask][i] will be the minimum cost to visit all cities in mask ending at city i
    dp = [[float('inf')] * n for _ in range(1 << n)]
    dp[1][0] = 0  # Starting at city 0

    # Iterate over all possible subsets of cities
    for mask in range(1 << n):
        for u in range(n):
            if mask & (1 << u):  # If u is in the subset represented by mask
                for v in range(n):
                    if not mask & (1 << v):  # If v is not in the subset represented by mask
                        new_mask = mask | (1 << v)
                        dp[new_mask][v] = min(dp[new_mask][v], dp[mask][u] + graph[u][v])

    # Find the minimum cost to visit all cities and return to the starting city
    res = float('inf')
    final_mask = (1 << n) - 1
    for u in range(1, n):
        res = min(res, dp[final_mask][u] + graph[u][0])

    return res

# Example usage
graph = [
    [0, 10, 15, 20],
    [10, 0, 35, 25],
    [15, 35, 0, 30],
    [20, 25, 30, 0]
]

print(tsp(graph))  # Output: 80
```

Output:



Time complexity:
O(n.k)