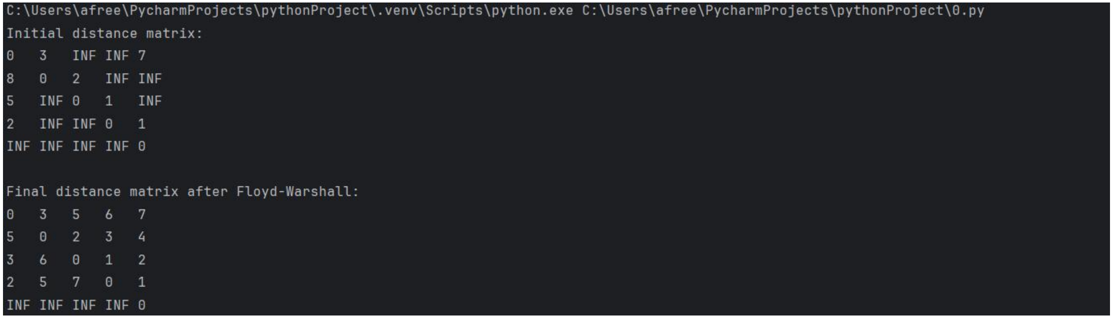
Exercise172:-
12. Implement Floyd's Algorithm to find the shortest path between all pairs of cities. Display the distance matrix before and after applying the algorithm. Identify and print the shortest path

```
Program:-import sys
def floyd_warshall(graph):
    num_vertices = len(graph)
    dist = [[sys.maxsize] * num_vertices for _ in range(num_vertices)]
    next_node = [[None] * num_vertices for _ in range(num_vertices)]
    for i in range(num_vertices):
        for j in range(num_vertices):
            if i == j:
                dist[i][j] = 0
            elif graph[i][j] != 0:
                dist[i][j] = graph[i][j]
                next_node[i][j] = j
    print("Initial distance matrix:")
    print_matrix(dist)
    for k in range(num_vertices):
        for i in range(num_vertices):
            for j in range(num_vertices):
                if dist[i][k] != sys.maxsize and dist[k][j] != sys.maxsize and dist[i][k] + dist[k][j] < dist[i][j]:
                    dist[i][j] = dist[i][k] + dist[k][j]
                    next_node[i][j] = next_node[i][k]
    print("\nFinal distance matrix after Floyd-Warshall:")
    print_matrix(dist)
    print("\nShortest paths between all pairs of cities:")
    for i in range(num_vertices):
        for j in range(num_vertices):
            if i != j:
                path = get_path(i, j, next_node)
                if path:
                    print(f"Shortest path from {i} to {j}: {path} with distance {dist[i][j]}")
    return dist
def get_path(u, v, next_node):
    if next_node[u][v] is None:
        return []
    path = [u]
    while u != v:
        u = next_node[u][v]
        path.append(u)
    return path
def print_matrix(matrix):
    for row in matrix:
        for val in row:
            if val == sys.maxsize:
                print("INF", end="\t")
            else:
                print(val, end="\t")
        print()
graph = [
    [0, 3, sys.maxsize, sys.maxsize, 7],
    [8, 0, 2, sys.maxsize, sys.maxsize],
    [5, sys.maxsize, 0, 1, sys.maxsize],
    [2, sys.maxsize, sys.maxsize, 0, 1],
    [sys.maxsize, sys.maxsize, sys.maxsize, sys.maxsize, 0]
]
```

```
floyd_warshall(graph)
```

Output:-



```
C:\Users\afree\PycharmProjects\pythonProject\.venv\Scripts\python.exe C:\Users\afree\PycharmProjects\pythonProject\0.py
Initial distance matrix:
0   3   INF INF 7
8   0   2   INF INF
5   INF 0   1   INF
2   INF INF 0   1
INF INF INF INF 0

Final distance matrix after Floyd-Warshall:
0   3   5   6   7
5   0   2   3   4
3   6   0   1   2
2   5   7   0   1
INF INF INF INF 0
```

Time complexity:-O(v^3)