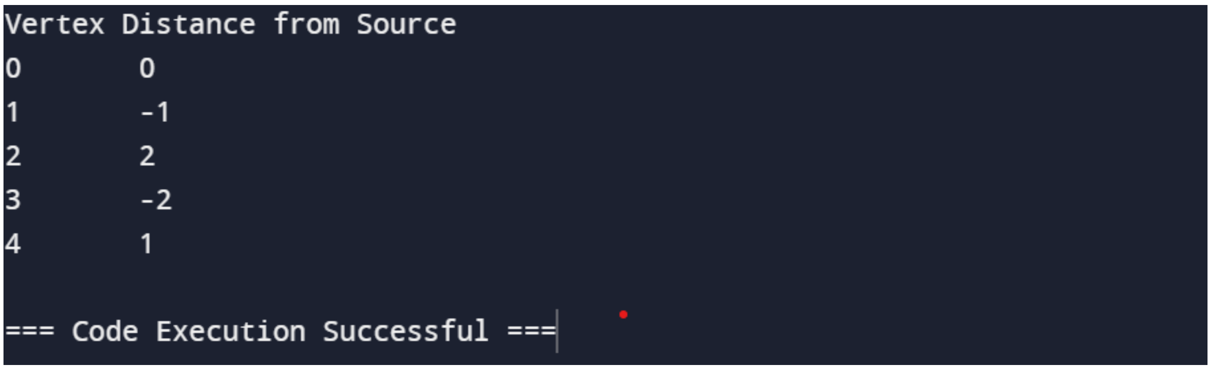
Exersice:- 110 Bellman Ford algorithm

Program:-

```
class Graph:
    def __init__(self, vertices):
        self.V = vertices  # Number of vertices
        self.edges = []    # List of edges
    def add_edge(self, u, v, w):
        self.edges.append((u, v, w))
    def bellman_ford(self, src):
        dist = [float('inf')] * self.V
        dist[src] = 0
        for _ in range(self.V - 1):
            for u, v, w in self.edges:
                if dist[u] != float('inf') and dist[u] + w < dist[v]:
                    dist[v] = dist[u] + w
        for u, v, w in self.edges:
            if dist[u] != float('inf') and dist[u] + w < dist[v]:
                return "Graph contains a negative weight cycle"
        return dist
if __name__ == "__main__":
    g = Graph(5)
    g.add_edge(0, 1, -1)
    g.add_edge(0, 2, 4)
    g.add_edge(1, 2, 3)
    g.add_edge(1, 3, 2)
    g.add_edge(1, 4, 2)
    g.add_edge(3, 2, 5)
    g.add_edge(3, 1, 1)
    g.add_edge(4, 3, -3)
    result = g.bellman_ford(0)
    if isinstance(result, str):
        print(result)
    else:
        print("Vertex Distance from Source")
        for i in range(len(result)):
            print(f"{i}\t\t{result[i]}")
```

output:-



```
Vertex Distance from Source
0       0
1       -1
2       2
3       -2
4       1

=== Code Execution Successful ===
```

Time complexity:-O(V.E)