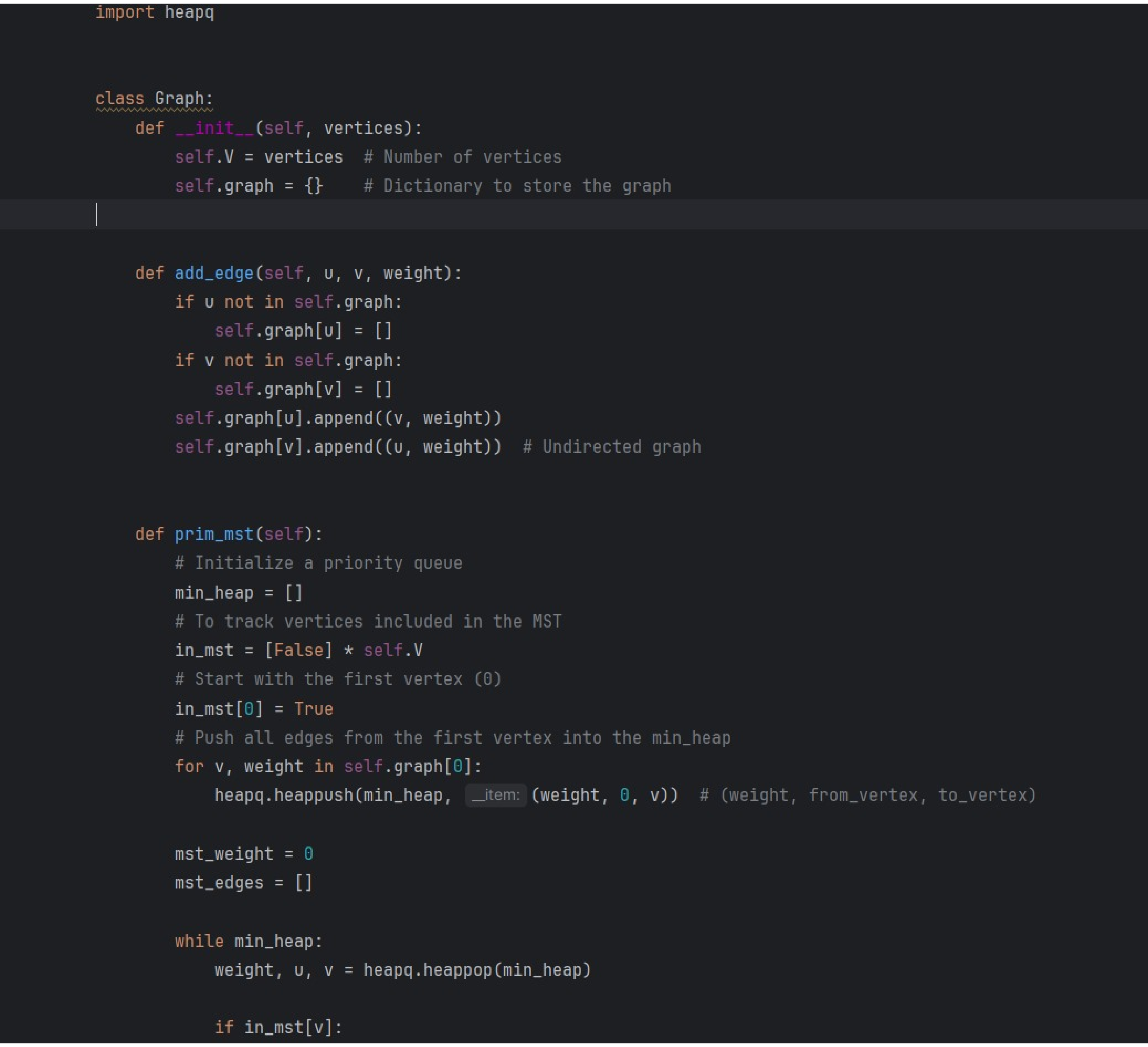
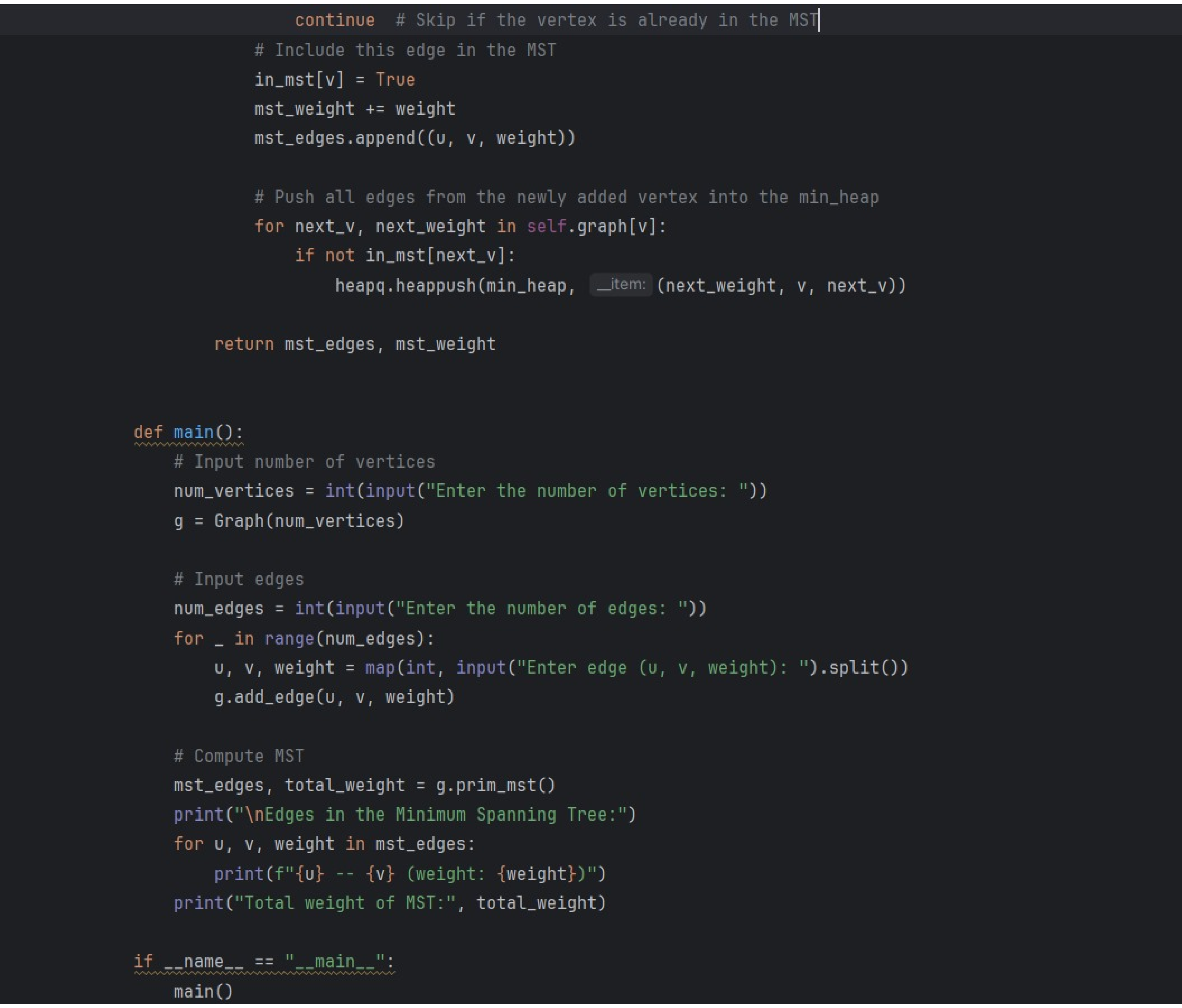
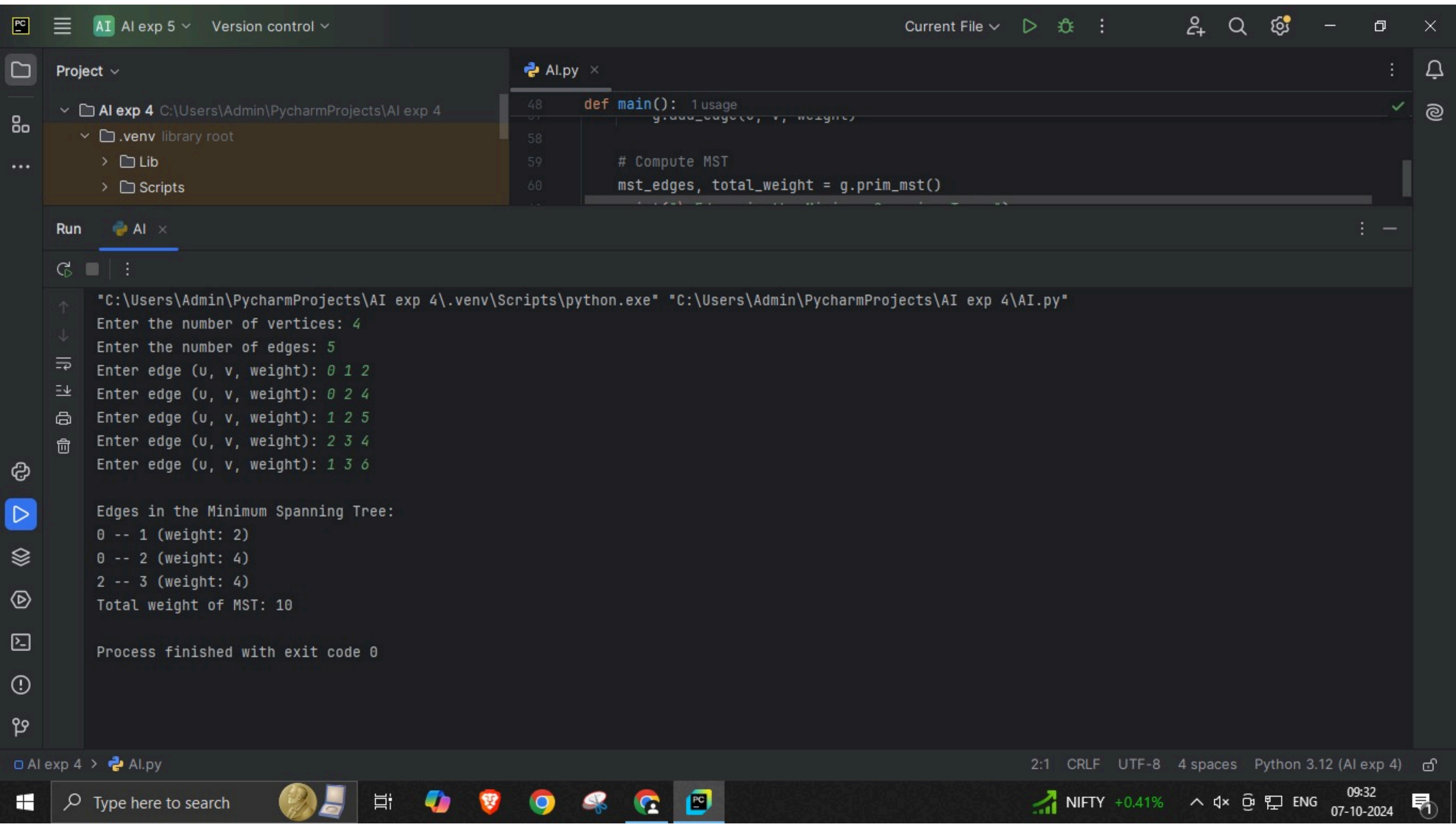


```python
import heapq


class Graph:
    def __init__(self, vertices):
        self.V = vertices  # Number of vertices
        self.graph = {}    # Dictionary to store the graph


    def add_edge(self, u, v, weight):
        if u not in self.graph:
            self.graph[u] = []
        if v not in self.graph:
            self.graph[v] = []
        self.graph[u].append((v, weight))
        self.graph[v].append((u, weight))  # Undirected graph


    def prim_mst(self):
        # Initialize a priority queue
        min_heap = []
        # To track vertices included in the MST
        in_mst = [False] * self.V
        # Start with the first vertex (0)
        in_mst[0] = True
        # Push all edges from the first vertex into the min_heap
        for v, weight in self.graph[0]:
            heapq.heappush(min_heap, __item: (weight, 0, v))  # (weight, from_vertex, to_vertex)

        mst_weight = 0
        mst_edges = []

        while min_heap:
            weight, u, v = heapq.heappop(min_heap)

            if in_mst[v]:
```



```python
                continue  # Skip if the vertex is already in the MST
            # Include this edge in the MST
            in_mst[v] = True
            mst_weight += weight
            mst_edges.append((u, v, weight))

            # Push all edges from the newly added vertex into the min_heap
            for next_v, next_weight in self.graph[v]:
                if not in_mst[next_v]:
                    heapq.heappush(min_heap,  __item: (next_weight, v, next_v))

        return mst_edges, mst_weight


def main():
    # Input number of vertices
    num_vertices = int(input("Enter the number of vertices: "))
    g = Graph(num_vertices)

    # Input edges
    num_edges = int(input("Enter the number of edges: "))
    for _ in range(num_edges):
        u, v, weight = map(int, input("Enter edge (u, v, weight): ").split())
        g.add_edge(u, v, weight)

    # Compute MST
    mst_edges, total_weight = g.prim_mst()
    print("\nEdges in the Minimum Spanning Tree:")
    for u, v, weight in mst_edges:
        print(f"{u} -- {v} (weight: {weight})")
    print("Total weight of MST:", total_weight)

if __name__ == "__main__":
    main()
```



```python
def main():  1 usage
        g.add_edge(u, v, weight)

    # Compute MST
    mst_edges, total_weight = g.prim_mst()
```

```
"C:\Users\Admin\PycharmProjects\AI exp 4\.venv\Scripts\python.exe" "C:\Users\Admin\PycharmProjects\AI exp 4\AI.py"
Enter the number of vertices: 4
Enter the number of edges: 5
Enter edge (u, v, weight): 0 1 2
Enter edge (u, v, weight): 0 2 4
Enter edge (u, v, weight): 1 2 5
Enter edge (u, v, weight): 2 3 4
Enter edge (u, v, weight): 1 3 6

Edges in the Minimum Spanning Tree:
0 -- 1 (weight: 2)
0 -- 2 (weight: 4)
2 -- 3 (weight: 4)
Total weight of MST: 10

Process finished with exit code 0
```