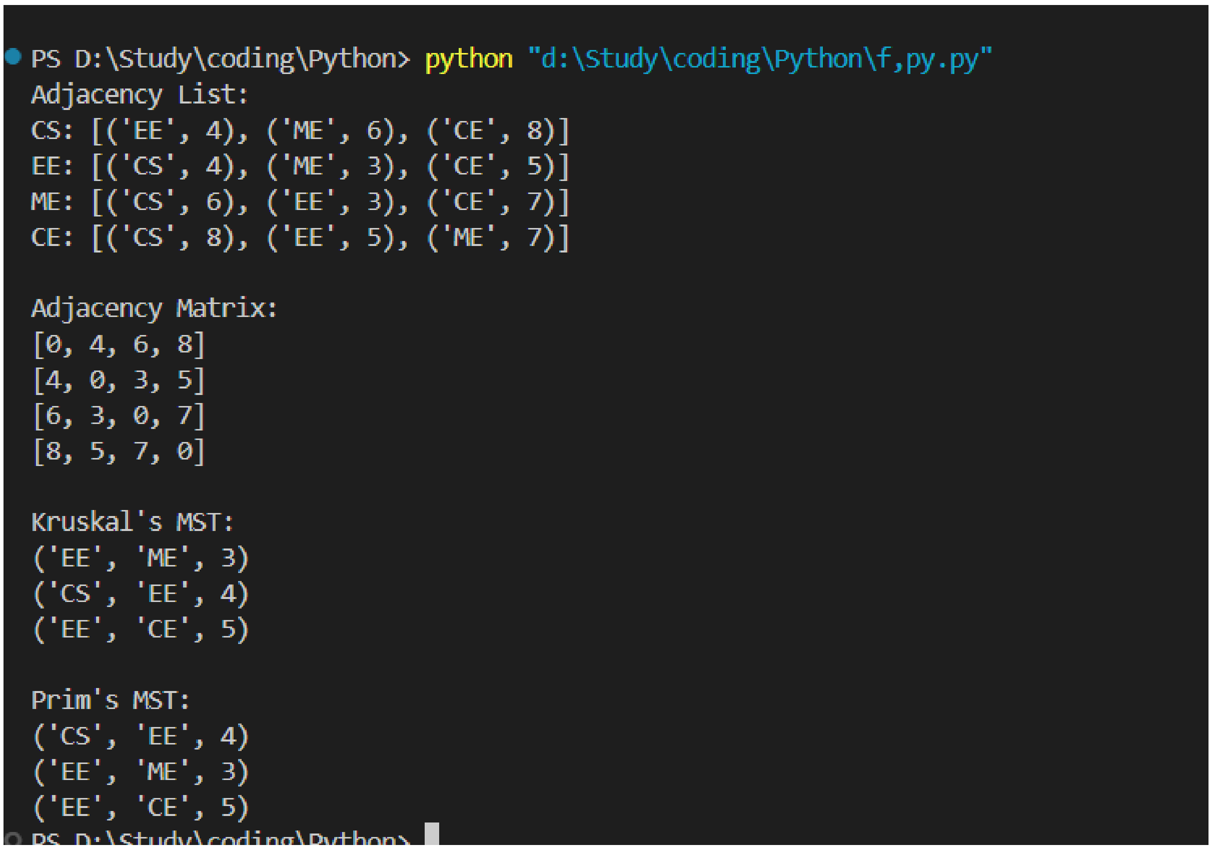
CODE:

```
import networkx as nx
import matplotlib.pyplot as plt

# --- Graph Setup ---
departments = ['CS', 'EE', 'ME', 'CE']
edges = [
   ('CS', 'EE', 4),
   ('CS', 'ME', 6),
   ('CS', 'CE', 8),
   ('EE', 'ME', 3),
   ('EE', 'CE', 5),
   ('ME', 'CE', 7)
]

# Build index map for matrix use
index = {dept: i for i, dept in enumerate(departments)}
n = len(departments)

# Adjacency List
adj_list = {dept: [] for dept in departments}
for src, dest, dist in edges:
   adj_list[src].append((dest, dist))
   adj_list[dest].append((src, dist))

print("Adjacency List:")
for k, v in adj_list.items():
   print(f"{k}: {v}")

# Adjacency Matrix
adj_matrix = [[0]*n for _ in range(n)]
for src, dest, dist in edges:
   i, j = index[src], index[dest]
   adj_matrix[i][j] = dist
   adj_matrix[j][i] = dist
print("\nAdjacency Matrix:")
for row in adj_matrix:
   print(row)

# --- Kruskal's Algorithm ---
def kruskal(departments, edges):
   parent = {dept: dept for dept in departments}
   def find(u):
      while parent[u] != u:
            parent[u] = parent[parent[u]]
            u = parent[u]
        return u
    def union(u, v):
        parent[find(u)] = find(v)
    mst = []
    edges_sorted = sorted(edges, key=lambda x: x[2])
    for u, v, w in edges_sorted:
        if find(u) != find(v):
            union(u, v)
            mst.append((u, v, w))
    return mst

mst_kruskal = kruskal(departments, edges)
print("\nKruskal's MST:")
for edge in mst_kruskal:
    print(edge)

# --- Prim's Algorithm ---
def prim_simple(departments, adj_list):
    start = departments[0]
    visited = set([start])
    mst = []
    while len(visited) < len(departments):
        min_edge = None
        for src in visited:
            for dest, dist in adj_list[src]:
                if dest not in visited:
                    if min_edge is None or dist < min_edge[2]:
                        min_edge = (src, dest, dist)
        if min_edge:
            _, dest, _ = min_edge
            visited.add(dest)
            mst.append(min_edge)
    return mst

mst_prim = prim(departments, adj_list)
print("\nPrim's MST:")
for edge in mst_prim:
    print(edge)

# --- Visualization Function ---
def draw_graph(nodes, edges, title):
    G = nx.Graph()
    G.add_weighted_edges_from(edges)
    pos = nx.spring_layout(G)  # nice layout
    labels = nx.get_edge_attributes(G, 'weight')

    plt.figure(figsize=(6,4))
    nx.draw(G, pos, with_labels=True, node_color='lightblue',
            node_size=1200, font_size=10, font_weight='bold', edgecolors='black')
    nx.draw_networkx_edge_labels(G, pos, edge_labels=labels)
    plt.title(title)
    plt.show()

# --- Display Graphs ---
draw_graph(departments, edges, "Original Campus Graph")
draw_graph(departments, mst_kruskal, "MST by Kruskal")
draw_graph(departments, mst_prim, "MST by Prim")
```



```
PS D:\Study\coding\Python> python "d:\Study\coding\Python\f,py.py"
Adjacency List:
CS: [('EE', 4), ('ME', 6), ('CE', 8)]
EE: [('CS', 4), ('ME', 3), ('CE', 5)]
ME: [('CS', 6), ('EE', 3), ('CE', 7)]
CE: [('CS', 8), ('EE', 5), ('ME', 7)]

Adjacency Matrix:
[0, 4, 6, 8]
[4, 0, 3, 5]
[6, 3, 0, 7]
[8, 5, 7, 0]

Kruskal's MST:
('EE', 'ME', 3)
('CS', 'EE', 4)
('EE', 'CE', 5)

Prim's MST:
('CS', 'EE', 4)
('EE', 'ME', 3)
('EE', 'CE', 5)
PS D:\Study\coding\Python>
```

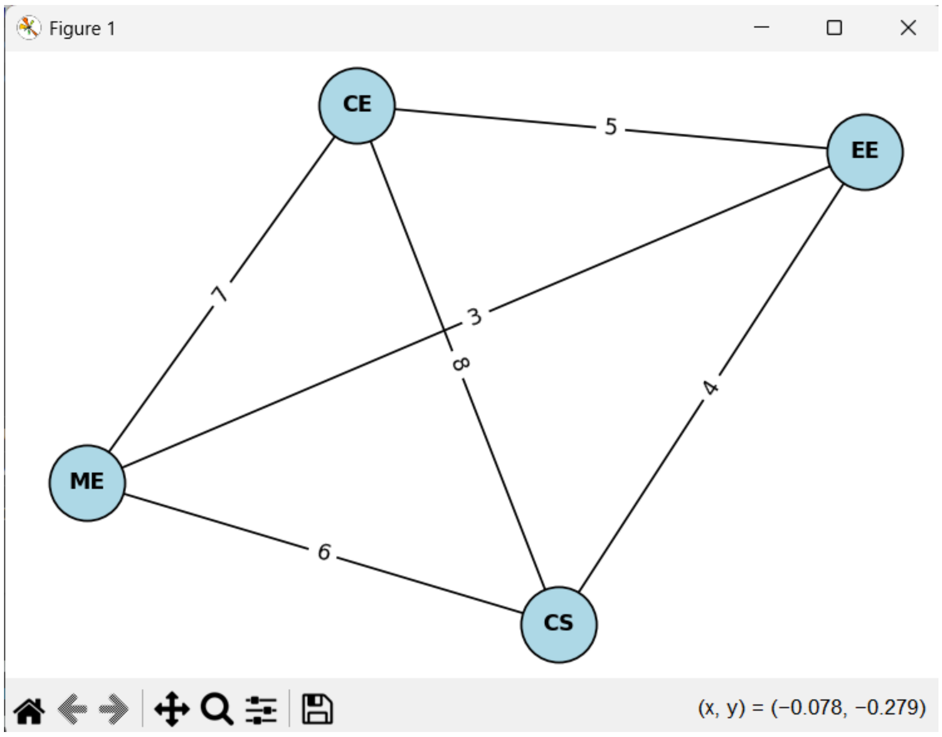


Figure 1
CE
EE
ME
CS
5
7
3
8
4
6
(x, y) = (−0.078, −0.279)

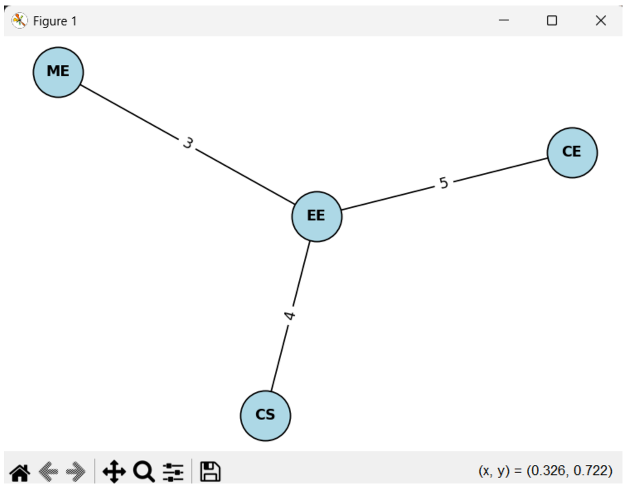


Figure 1
ME
CE
EE
CS
3
5
4
(x, y) = (0.326, 0.722)

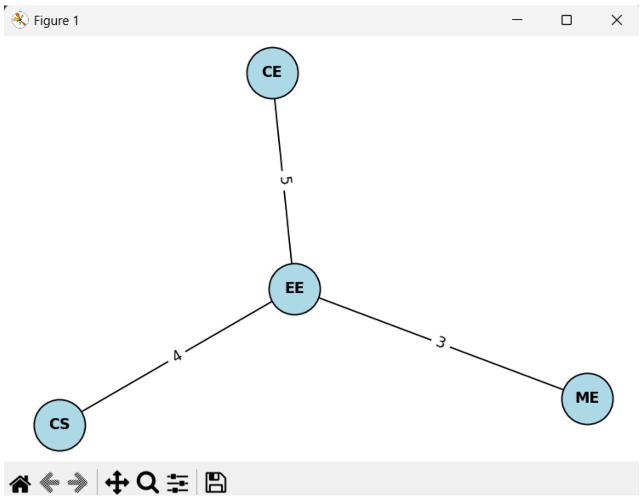


Figure 1
CE
EE
CS
ME
5
4
3