Create an empty graph G.

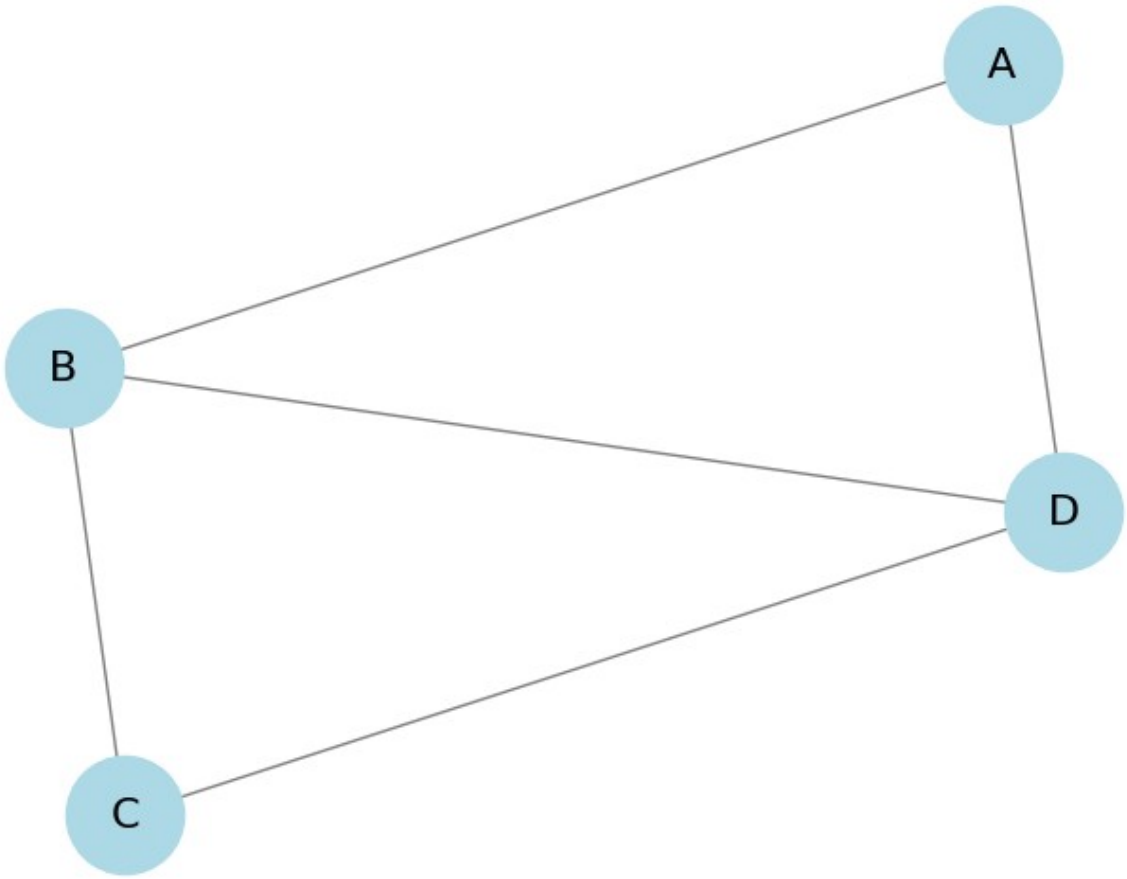
Add the following nodes: A, B, C, D

Add the following edges: A-B, B-C, C-D, D-A, B-D

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

# Creating an empty graph
G = nx.Graph()

# Adding the given nodes to the graph
G.add_nodes_from(["A", "B", "C", "D"])

# Adding the given edges to the graph
G.add_edges_from([("A", "B"), ("B", "C"), ("C", "D"), ("D", "A"),
("B", "D")])
```

Visualize the Graph

```
# Drawing the graph
nx.draw(G, with_labels=True, node_color='lightblue',
edge_color='gray', node_size=2000, font_size=16)
plt.show()
```



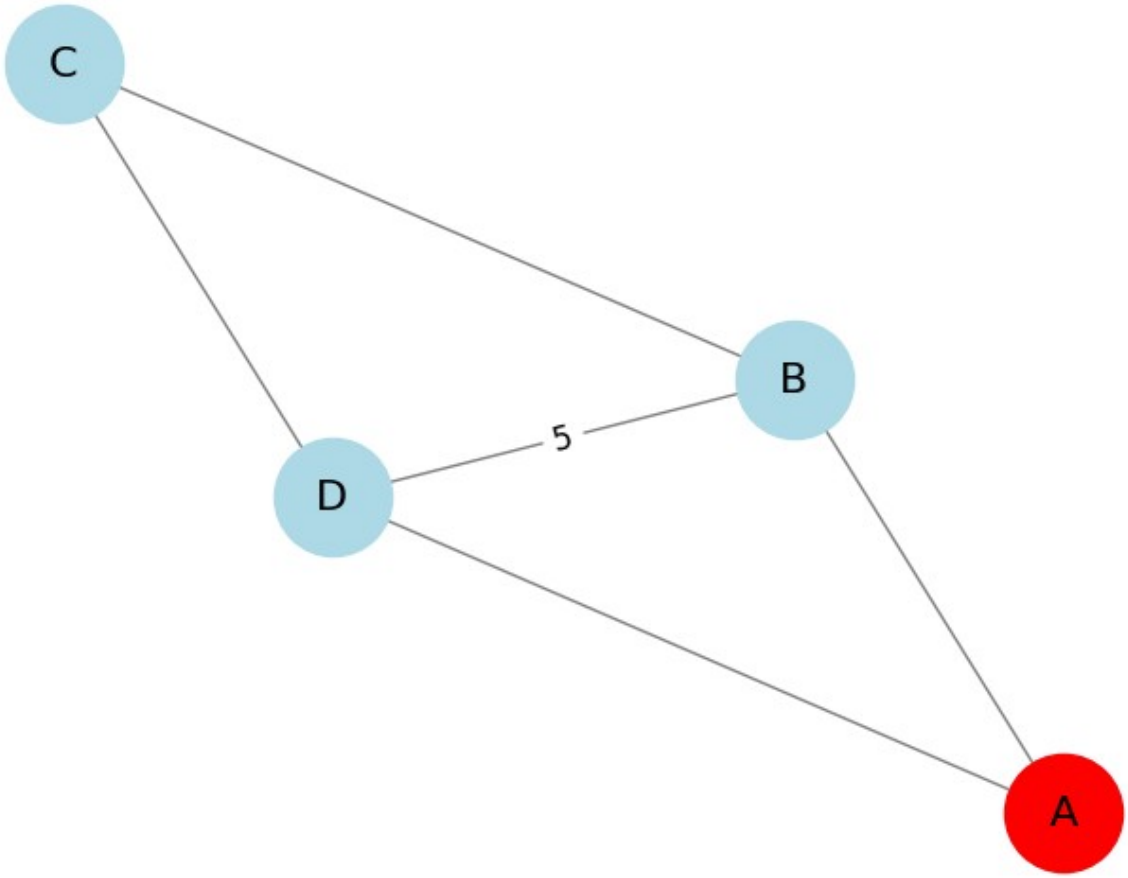
Add an attribute color to node A with value 'red'.

Add a weight attribute to edge (B, D) with value 5.

```
# Adding attributes to the graph
G.nodes["A"]["color"] = "red"  # Adding the color attribute to node A
G.edges["B", "D"]["weight"] = 5  # Adding the weight attribute to edge
(B, D)

# Printing the attributes
print("Node A attributes:", G.nodes["A"])
print("Edge (B, D) attributes:", G.edges["B", "D"])

# Drawing the graph
pos = nx.spring_layout(G)  # This is the layout for visualization
nx.draw(G, pos, with_labels=True, node_color='lightblue',
edge_color='gray', node_size=2000, font_size=16)

# Highlighting node A
nx.draw_networkx_nodes(G, pos, nodelist=["A"], node_color="red",
node_size=2000)
```

```
# Showing the edge labels (weights)
edge_labels = {("B", "D"): "5"}
nx.draw_networkx_edge_labels(G, pos, edge_labels=edge_labels,
font_size=12)

plt.show()
```

```
Node A attributes: {'color': 'red'}
Edge (B, D) attributes: {'weight': 5}
```



How many nodes are there in the graph?

How many edges are there?

List all the neighbors of node B.

```
# Geting the number of nodes
num_nodes = G.number_of_nodes()
print("Number of nodes:", num_nodes)

# Geting the number of edges
num_edges = G.number_of_edges()
```

```
print("Number of edges:", num_edges)

# Listing out all neighbors of node B
neighbors_B = list(G.neighbors("B"))
print("Neighbors of node B:", neighbors_B)
```

```
Number of nodes: 4
Number of edges: 5
Neighbors of node B: ['A', 'C', 'D']
```

Find the shortest path between node A and node C (assuming unweighted edges).

```
# Finding the shortest path from A to C
shortest_path_A_C = nx.shortest_path(G, source="A", target="C")
print("Shortest path from A to C:", shortest_path_A_C)
```

```
Shortest path from A to C: ['A', 'B', 'C']
```