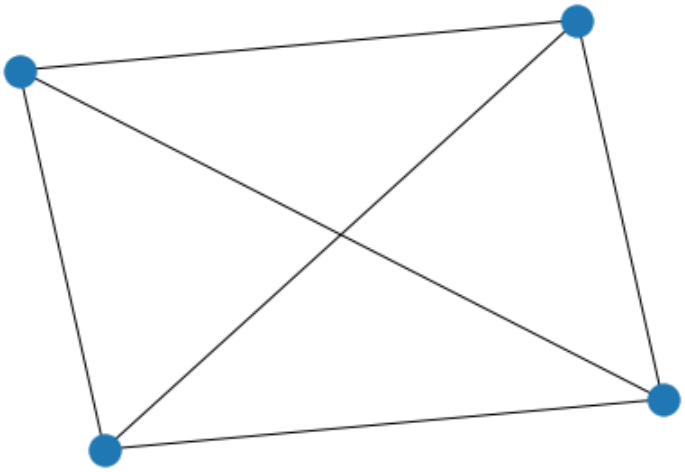
Exercise 1 - Theory

In [ ]:
```
import networkx as nx
from matplotlib import pyplot as plt
```

Problem 1 - Königsberg Problem

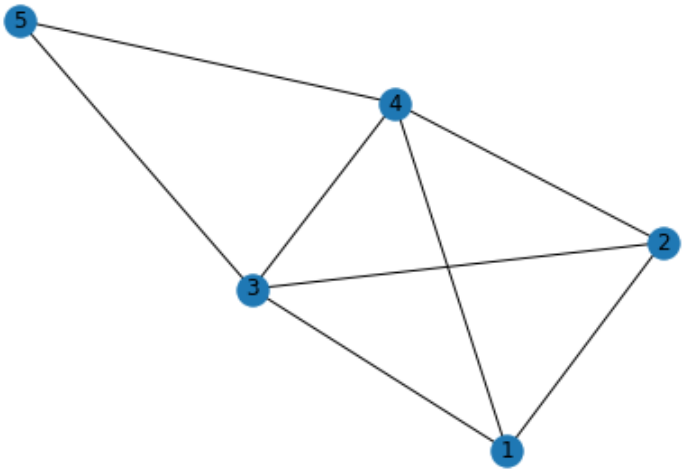
a) Which of the shown graphs can be drawn without raising the pencil from the paper and without drawing one line more than once. Why? (2 pts)

In [ ]:
```
G = nx.Graph()
G.add_edges_from([(1,2), (1,3), (1,4), (2,3), (2,4), (3,4)])
nx.draw(G)
```



Not possible to trace since all 4 vertices have an odd number of edges but one has to either begin or stop at an odd vertex.

In [ ]:
```
G = nx.Graph()
G.add_edges_from([(1,2), (1,3), (1,4), (2,3), (2,4), (3,4), (3,5), (4,5)])
nx.draw(G, with_labels = True)
```

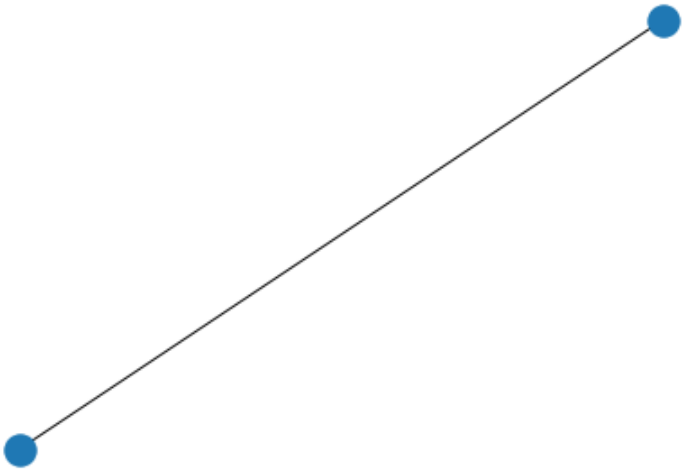


Here it is possible to trace it since there are exactly 2 odd vertices. One could start at node 1 and end at node 2 or the other way around.

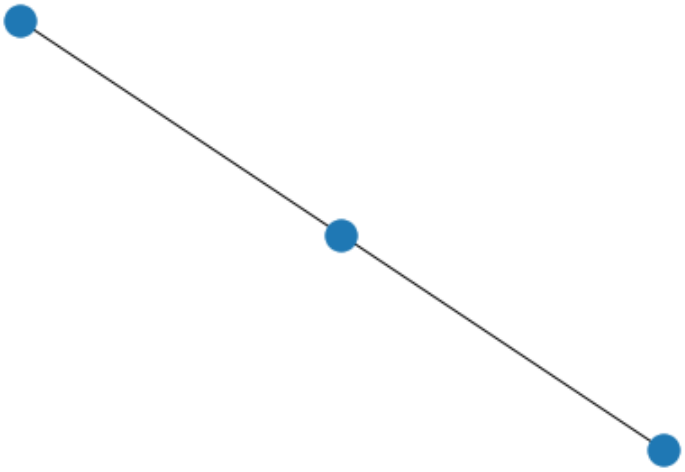
b) Find two graphs with and two without having this property. (2 pts)

examples of traceable graphs:

In [ ]:
```
G1 = nx.Graph()
G1.add_edges_from([(1,2)])
nx.draw(G1)
```

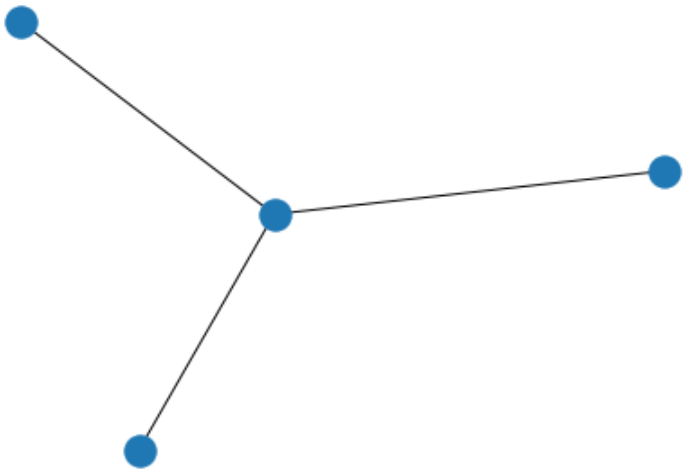


In [ ]:
```
G2 = nx.Graph()
G2.add_edges_from([(1,2), (2,3)])
nx.draw(G2)
```

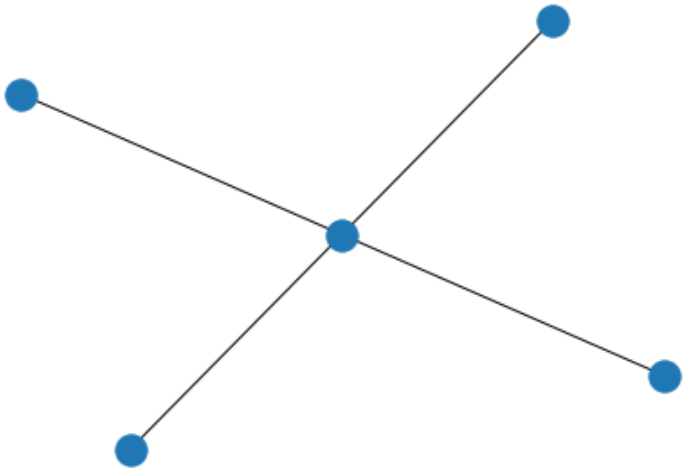


examples of non-traceable graphs:

In [ ]:
```
G1 = nx.Graph()
G1.add_edges_from([(1,2), (1,3), (1,4)])
nx.draw(G1)
```



In [ ]:
```
G2 = nx.Graph()
G2.add_edges_from([(1,2), (1,3), (1,4), (1,5)])
nx.draw(G2)
```

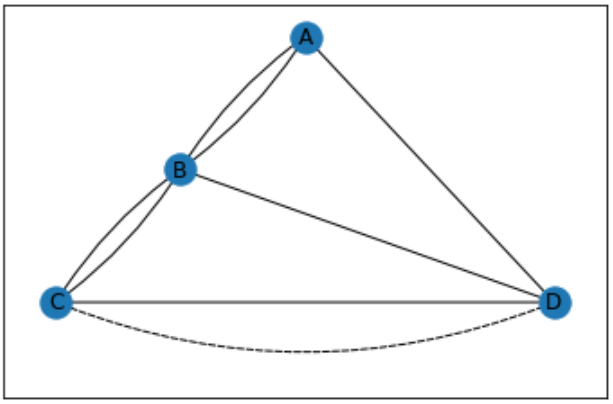


Problem 2 - Extended Königsberg Problem

a) A Baron living in the blue castle wants to start at his place and end up in the yellow bar on the island in the middle by walking all bridges once before. Where should he build the 8th bridge without enabling the Baroness in the red castle doing the same starting from her place? Find one path. (2 pts)

By building a bridge between baroness C and the other island D the only odd vertices left are the baron A and the bar B.

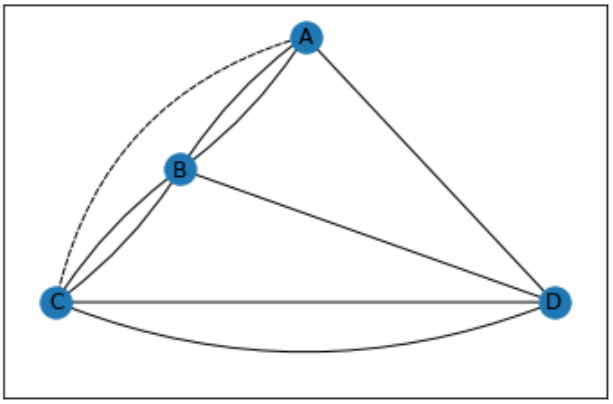
In [ ]:
```
G = nx.DiGraph()
G.add_nodes_from(["C","D","A","B"])

pos = nx.planar_layout(G)
nx.draw_networkx_nodes(G, pos)
nx.draw_networkx_edges(G, pos, edgelist = [("A","D"), ("B","D"), ("C","D")], arrowsty
nx.draw_networkx_edges(G, pos, edgelist = [("A","B"),("B","C")], arrowstyle = "-", co
nx.draw_networkx_edges(G, pos, edgelist = [("A","B"),("B","C")], arrowstyle = "-", co
nx.draw_networkx_edges(G, pos, edgelist = [("C","D")], arrowstyle = "-", style = "das
nx.draw_networkx_labels(G, pos)
plt.show()
```



b) The Baroness, on discovering this deception, comes up with her own plan to build a 9th bridge. It should enable her to walk from her castle to the bar using all bridges once, but it should make it now impossible for the blue Baron to do the same thing from his castle. Where should the 9th bridge go? Find one path. (2 pts)

By building a bridge between the baron A and the baroness C the only odd vertices are the baroness C and the bar B.

In [ ]:
```
G = nx.DiGraph()
G.add_nodes_from(["C","D","A","B"])

pos = nx.planar_layout(G)
nx.draw_networkx_nodes(G, pos)
nx.draw_networkx_edges(G, pos, edgelist = [("A","D"), ("B","D"), ("C","D")], arrowsty
nx.draw_networkx_edges(G, pos, edgelist = [("A","B"),("B","C")], arrowstyle = "-", co
nx.draw_networkx_edges(G, pos, edgelist = [("A","B"),("B","C")], arrowstyle = "-", co
nx.draw_networkx_edges(G, pos, edgelist = [("C","D")], arrowstyle = "-", connectionst
nx.draw_networkx_edges(G, pos, edgelist = [("A","C")], arrowstyle = "-", style = "das
nx.draw_networkx_labels(G, pos)
plt.show()
```

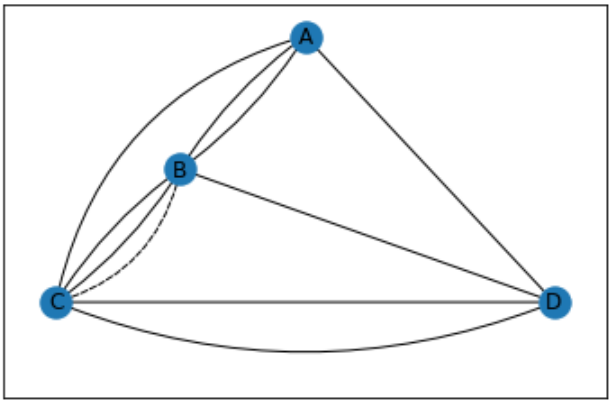


c) The mayor now decides to scupper their plans and builds a 10th bridge allowing all citizens starting from either side ending up in the bar by walking all the bridges. Where should it go? (1 pt)

Starting at any point and ending at any other point is not possible since there would need to be more than two odd vertices. But by building a bridge between the bar B and the baroness C we have that all vertices are even and thus one can start at any point and end up at the start using all bridges.

In [ ]:

```
G = nx.DiGraph()
G.add_nodes_from(["C","D","A","B"])

pos = nx.planar_layout(G)
nx.draw_networkx_nodes(G, pos)
nx.draw_networkx_edges(G, pos, edgelist = [("A","D"), ("B","D"), ("C","D")], arrowsty
nx.draw_networkx_edges(G, pos, edgelist = [("A","B"),("B","C")], arrowstyle = "-", co
nx.draw_networkx_edges(G, pos, edgelist = [("A","B"),("B","C")], arrowstyle = "-", co
nx.draw_networkx_edges(G, pos, edgelist = [("C","D")], arrowstyle = "-", connectionst
nx.draw_networkx_edges(G, pos, edgelist = [("A","C")], arrowstyle = "-", connectionst
nx.draw_networkx_edges(G, pos, edgelist = [("B","C")], arrowstyle = "-", style = "das
nx.draw_networkx_labels(G, pos)
plt.show()
```

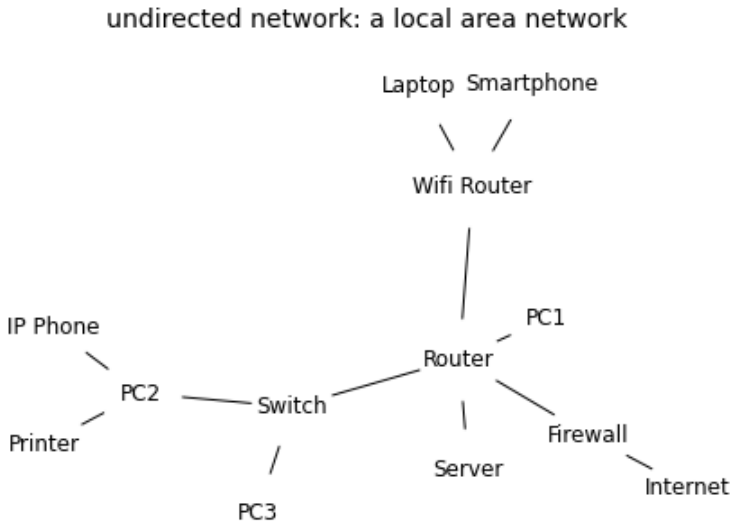


Problem 3

Find examples in nature, society or from your daily experience for an undirected network, a directed network and a network with edge weights. (3 pts)

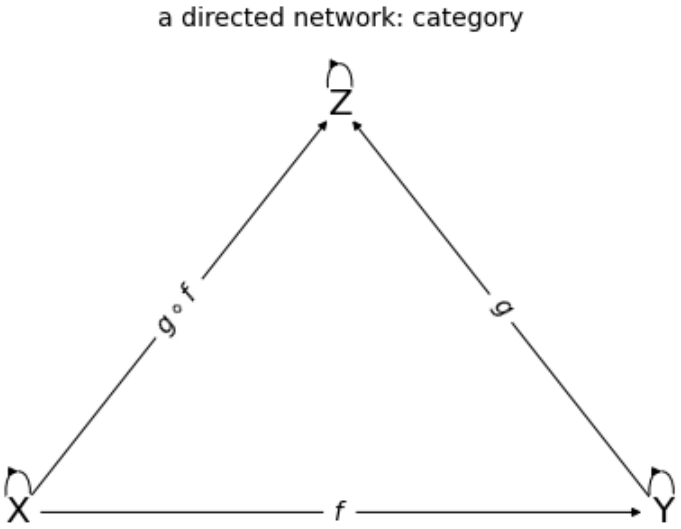
In [ ]:

```
G1 = nx.Graph()
G1.add_edges_from([
    ("Internet", "Firewall"),
    ("Router", "Firewall"),
    ("Router", "Switch"),
    ("Router", "Wifi Router"),
    ("Router", "Server"),
    ("Router", "PC1"),
    ("Switch", "PC2"),
    ("Switch", "PC3"),
    ("PC2", "Printer"),
    ("PC2", "IP Phone"),
    ("Wifi Router", "Smartphone"),
    ("Wifi Router", "Laptop"),
])

nx.draw(G1, with_labels = True, node_size = 2000, node_color = "w")
ax = plt.gca()
ax.set_title("undirected network: a local area network", fontsize = 14)
plt.show()
```



In [ ]:

```
G2 = nx.DiGraph()
X = "X"
Y = "Y"
Z = "Z"
G2.add_edges_from([(X,Y), (X,Z), (Y,Z)])

pos = nx.planar_layout(G2)
nx.draw(G2, pos, with_labels=True, node_size=500, node_color="w", font_size=20)
nx.draw_networkx_edges(G2, pos, edgelist=[(X,X), (Y,Y), (Z,Z)])
nx.draw_networkx_edge_labels(G2, pos, edge_labels={(X,Y):r"$f$", (Y,Z):r"$g$", (X,Z):
ax = plt.gca()
ax.set_title("a directed network: category", fontsize=14)
plt.show()
```



In [ ]:

```
G3 = nx.DiGraph()

G3.add_edge(1, 1, weight=0.5)
G3.add_edge(1, 2, weight=0.3)
G3.add_edge(1, 3, weight=0.2)
G3.add_edge(2, 1, weight=0.1)
G3.add_edge(2, 2, weight=0.5)
G3.add_edge(2, 3, weight=0.4)
G3.add_edge(3, 1, weight=0.7)
G3.add_edge(3, 2, weight=0.2)
G3.add_edge(3, 3, weight=0.1)

pos = nx.planar_layout(G3)
nx.draw(G3, pos, with_labels=True)
nx.draw_networkx_edge_labels(G3, pos, edge_labels=dict([((n1, n2), d['weight']) for r

ax = plt.gca()
ax.set_title("a weighted network: markov process", fontsize=14)
plt.show()
```