# 인공지능 10주차 출석과제

localhost:8888/notebooks/Documents/카카오톡 받은 파일/인공지능_10주차_출석과제.ipynb

In [1]:

```
import networkx as nx

# 빈 그래프 생성

G = nx.Graph()
```

In [2]:

```
# 빈 그래프에 노드 1개를 만들고 그 내용을 출력

G.add_node(1)

print(G.nodes())
```

```
[1]
```

In [3]:

```
# 숫자뿐만 아니라 문자나 문자열도 라벨로 사용 가능

G.add_node('P')

G.add_node('Hi')

print(G.nodes())
```

```
[1, 'P', 'Hi']
```

In [4]:

```
# 한번에 다수 개의 노드를 삽입 후 그 내용을 출력

G.add_nodes_from([2,3])

print(G.nodes())
```

```
[1, 'P', 'Hi', 2, 3]
```

In [5]:

```
# 지금까지 작성된 그래프 G에 입력된 노드들을 한 줄씩 출력

for node in G.nodes():

    print(node)
```

```
1
P
Hi
2
3
```

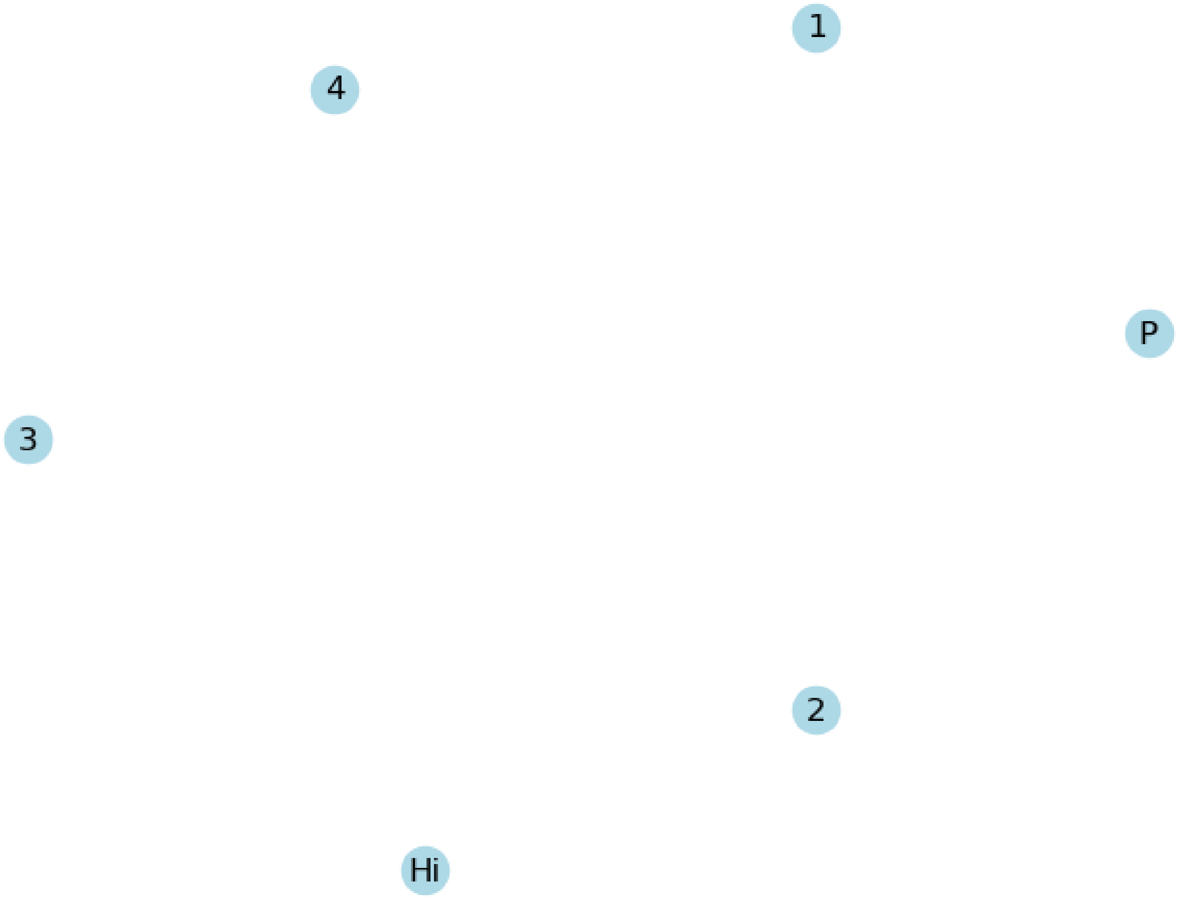
## In [6]:

```
G.add_nodes_from([3,4])

print(G.nodes())
```

```
[1, 'P', 'Hi', 2, 3, 4]
```

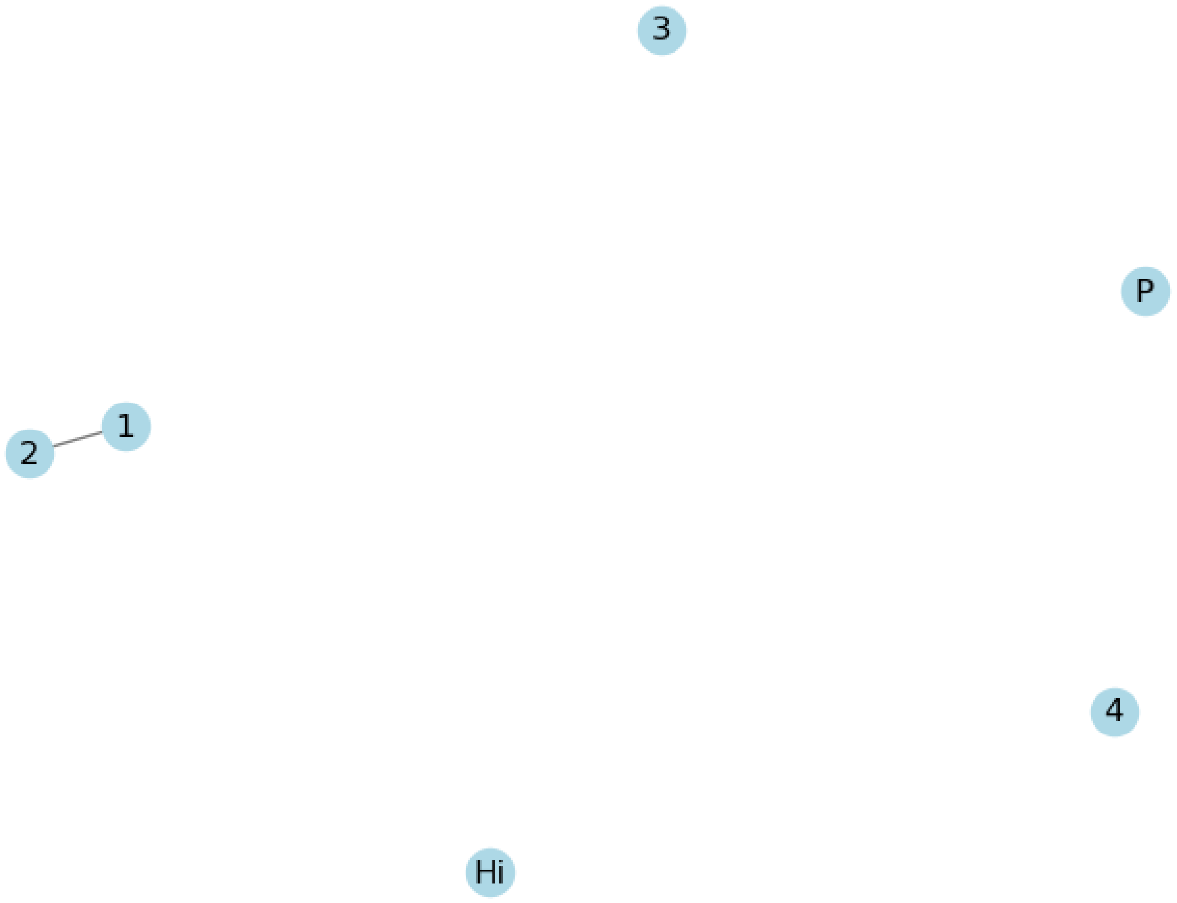
## In [7]:

```
nx.draw(G, with_labels=True, node_color='lightblue')
```



## In [8]:

```
# 엣지가 추가된 그래프

G.add_edge(1,2)

nx.draw(G, with_labels=True, node_color='lightblue',edge_color='grey')
```



In [9]:

```
# 그래프의 엣지 연결상태 출력

print(G.edges())

[(1, 2)]
```

In [10]:

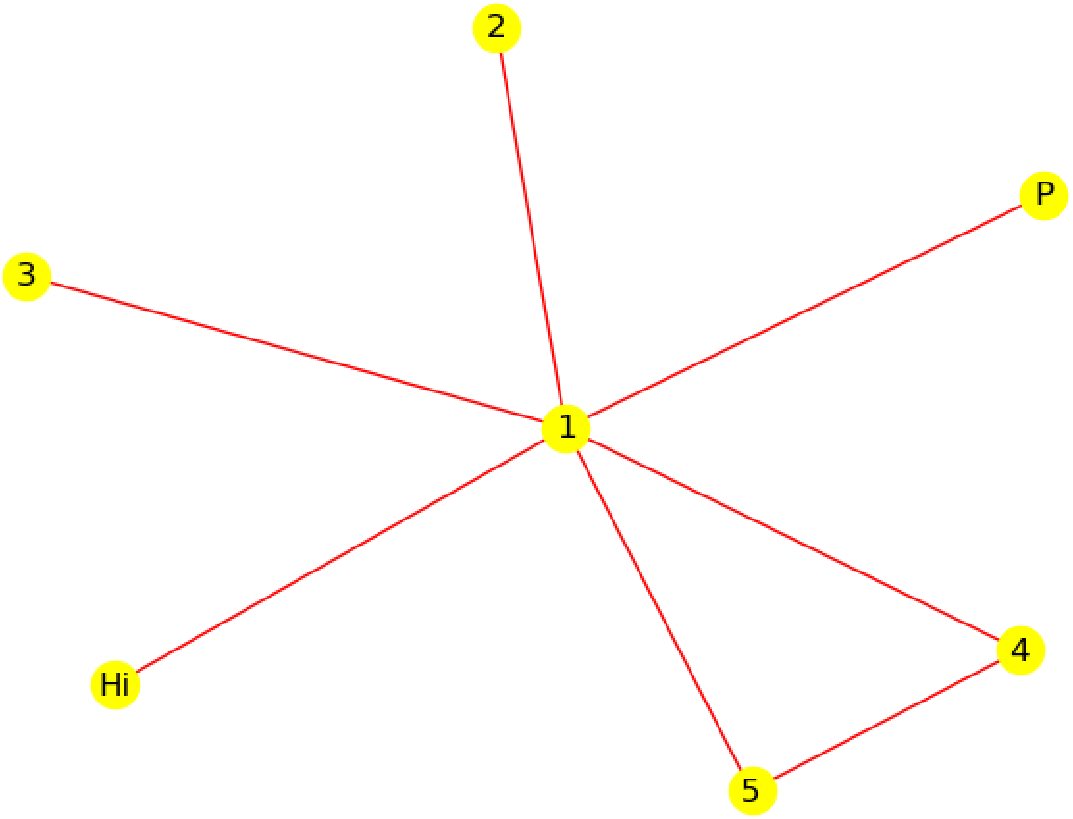
```
# 엣지를 연결할 때 기존에 없던 노드를 연결하는 경우

G.add_edge(4,5)

print(G.edges())

[(1, 2), (4, 5)]
```

In [11]:

```
# 복수 개의 엣지들을 한 번에 만들 수 있음

G.add_edges_from([(1,2),(1,3),(1,4),(1,5)])

print(G.edges())

[(1, 2), (1, 3), (1, 4), (1, 5), (4, 5)]
```

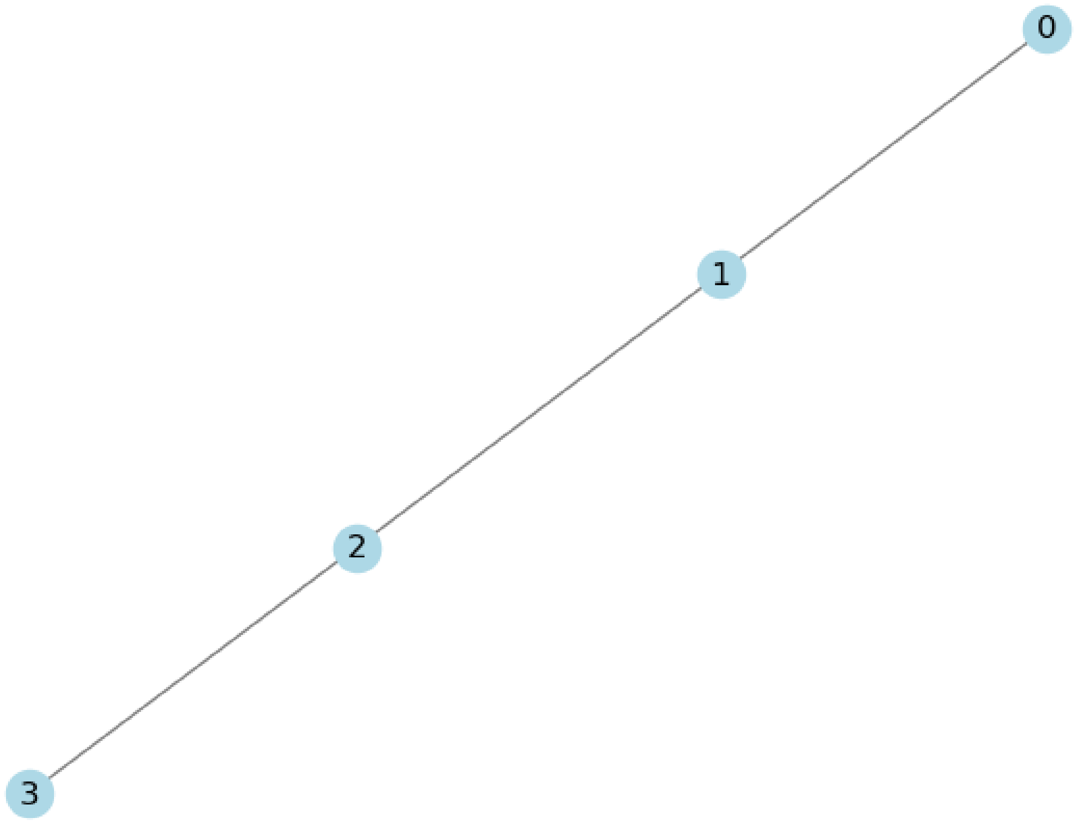
In [12]:

```
# 'P'와 'Hi'를 노드 1과 연결한 후 전체 그래프를 그리기
G.add_edge(1,'P')
G.add_edge(1,'Hi')
nx.draw(G, with_labels=True, node_color='yellow',edge_color='red')
```



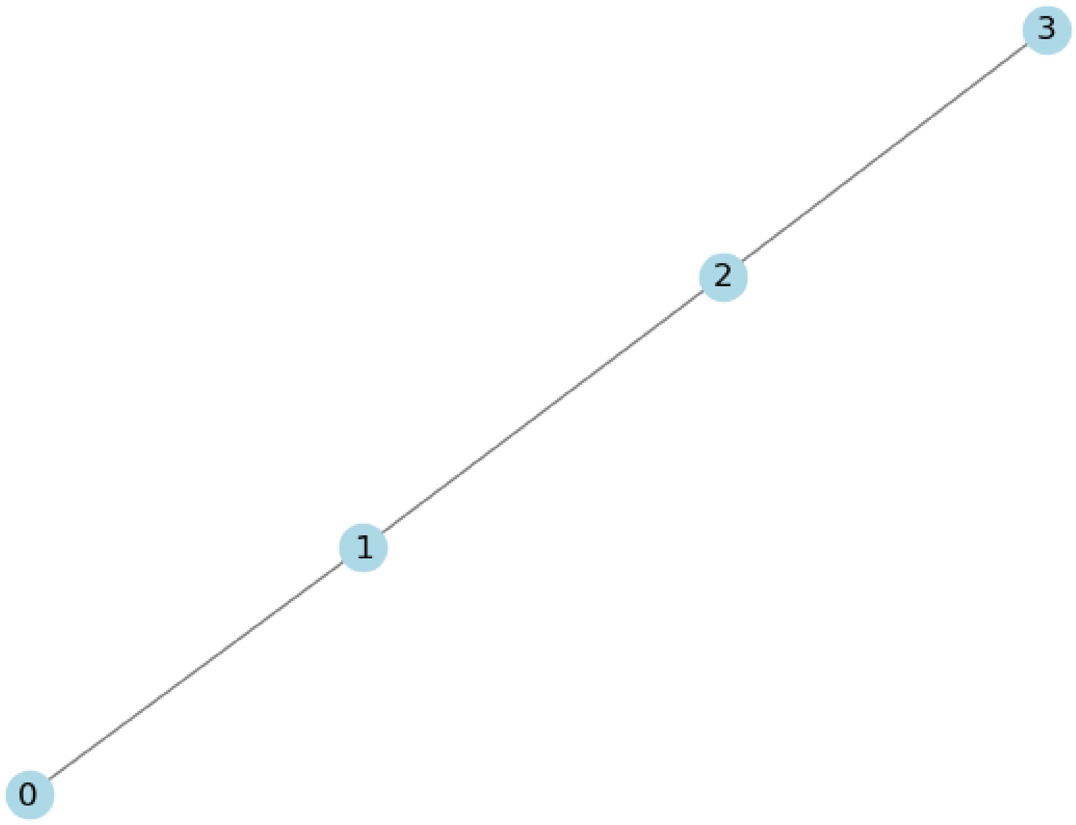
In [13]:

```
# 그래프 자동 생성 방법1
import networkx as nx

G = nx.path_graph(4)
nx.draw(G, with_labels=True, node_color='lightblue', edge_color='grey')
```



In [14]:

```
# 그래프 자동 생성 방법2

import networkx as nx


G = nx.Graph()

nx.add_path(G,[0,1,2,3])

nx.draw(G, with_labels=True, node_color='lightblue', edge_color='grey')
```



In [15]:

```
# 노드 별 차수 구하기

import networkx as nx



G = nx.Graph()

nx.add_path(G, [0,1,2,3])



print(G.degree(0))         # 노드 0에 대한 차수만 출력

print(G.degree([0, 1]))    # 노드 0과 노드 1에 대한 차수만 출력

print(G.degree())          # 모든 노드에 대한 차수를 출력
```

```
1
[(0, 1), (1, 2)]
[(0, 1), (1, 2), (2, 2), (3, 1)]
```

In [16]:

```
# 노드/엣지의 삭제

print(G.edges())
```

```
G.remove_edge(1, 2)

print(G.edges())
```

```
[(0, 1), (1, 2), (2, 3)]
[(0, 1), (2, 3)]
```

In [17]:

```
# 다수 개의 엣지 삭제

print(G.edges())

G.remove_edges_from([(0,1),(2,3)])

print(G.edges())
```

```
[(0, 1), (2, 3)]
[]
```

In [18]:

```
# 불필요한 노드 삭제하기

print(G.nodes())

G.remove_node(2)

print(G.nodes())
```

```
[0, 1, 2, 3]
[0, 1, 3]
```

In [19]:

```
# 다수 개의 노드를 한꺼번에 삭제하기

print(G.nodes())

G.remove_nodes_from([0, 1, 3])

print(G.nodes())
```

```
[0, 1, 3]
[]
```

In [20]:

```
# 노드/엣지의 갯수

import networkx as nx



G = nx.Graph()

G.add_nodes_from([1,2,3,4,5])
```

```
G.add_edges_from([(1,2), (1,3), (1,4), (1,5), (4,5)])



print('No. nodes:', G.number_of_nodes())

print('No. edges:', G.number_of_edges())
```

```
No. nodes: 5
No. edges: 5
```

In [21]:

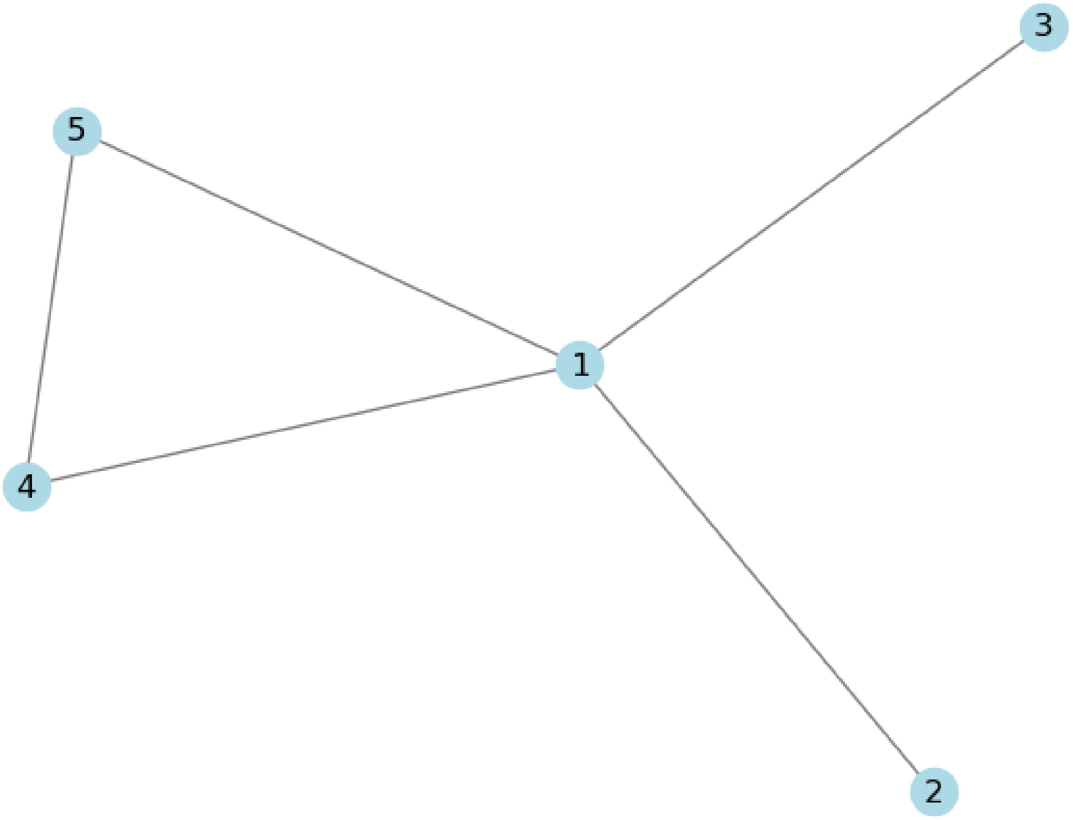
```
# 엣지 1개 제거

G.remove_edge(1,3)



print('No. nodes:', G.number_of_nodes())

print('No. edges:', G.number_of_edges())
```

```
No. nodes: 5
No. edges: 4
```

In [22]:

```
# 그래프 생성

import networkx as nx



G = nx.Graph()

G.add_nodes_from([1,2,3,4,5])

G.add_edges_from([(1,2),(1,3),(1,4),(1,5),(4,5)])



print('No. nodes:', G.number_of_nodes())

print('No. edges:', G.number_of_edges())



nx.draw(G, with_labels=True, node_color='lightblue', edge_color='grey')
```

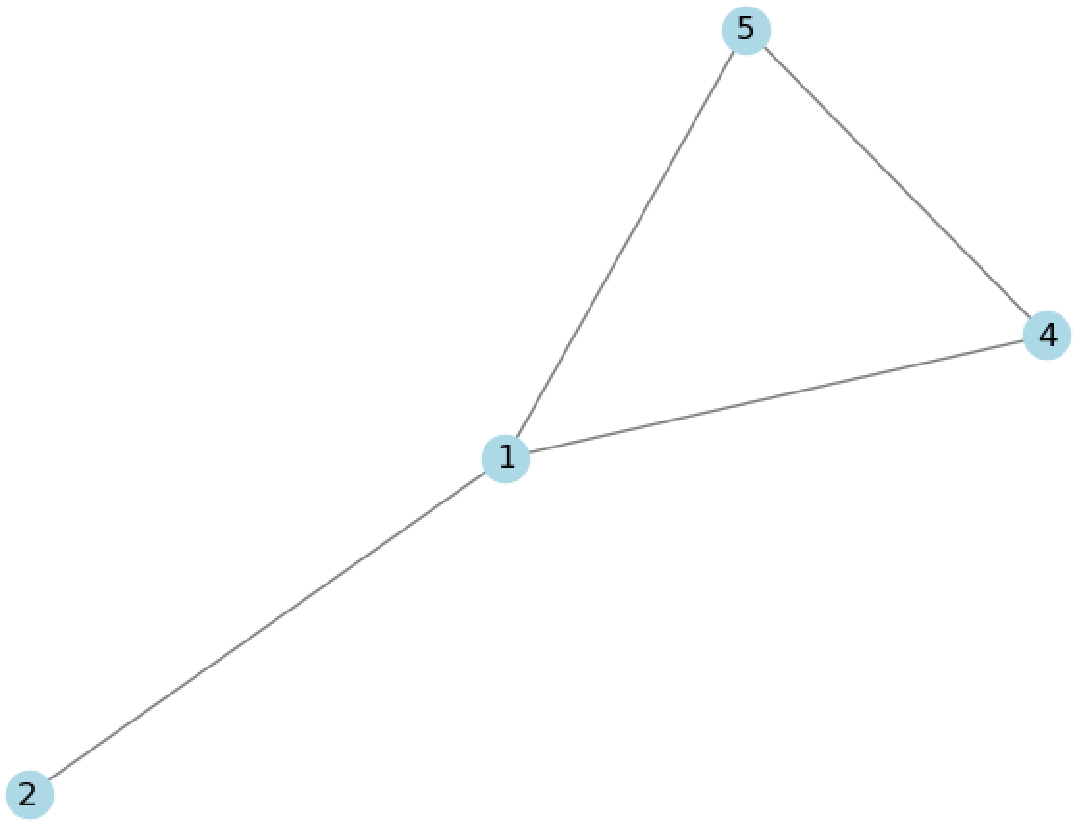
```
No. nodes: 5
No. edges: 5
```



In [23]:

```
# 노드 1개 제거; 연관된 엣지도 함께 제거

G.remove_node(3)



print('No. nodes:', G.number_of_nodes())

print('No. edges:', G.number_of_edges())



nx.draw(G, with_labels=True, node_color='lightblue', edge_color='grey')
```

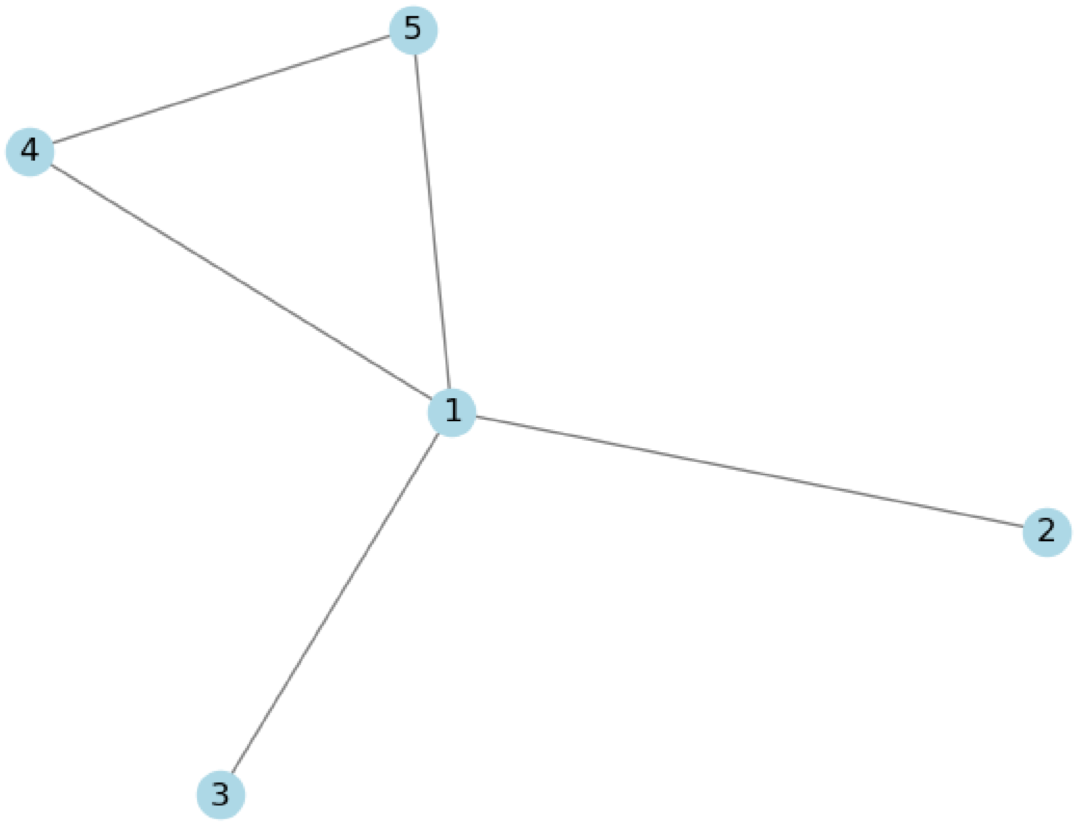
```
No. nodes: 4
No. edges: 4
```



In [24]:

```
# 그래프 생성

import networkx as nx


G = nx.Graph()

G.add_nodes_from([1,2,3,4,5])

G.add_edges_from([(1,2),(1,3),(1,4),(1,5),(4,5)])


print('No. nodes:', G.number_of_nodes())

print('No. edges:', G.number_of_edges())


nx.draw(G, with_labels=True, node_color='lightblue', edge_color='grey')
```

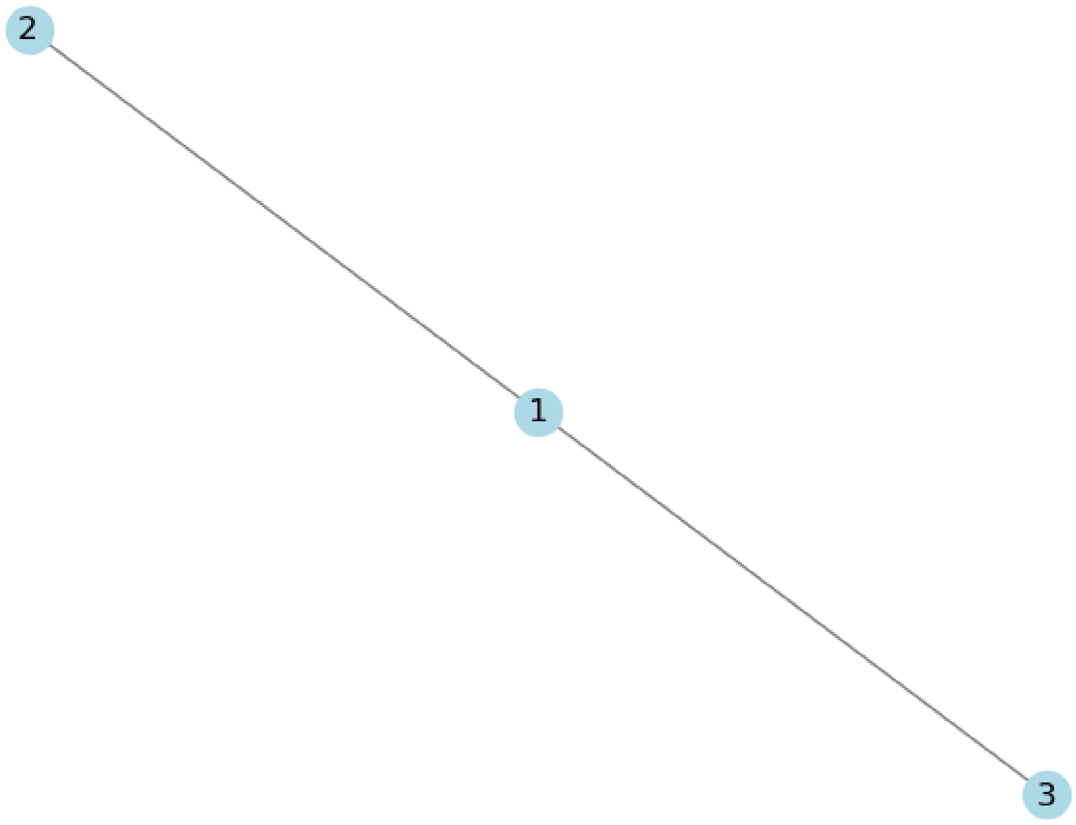
```
No. nodes: 5
No. edges: 5
```



In [25]:

```
# 복수 개의 노드 제거; 연관된 엣지도 함께 제거

G.remove_nodes_from([4,5])



print('No. nodes:', G.number_of_nodes())

print('No. edges:', G.number_of_edges())



nx.draw(G, with_labels=True, node_color='lightblue', edge_color='grey')
```

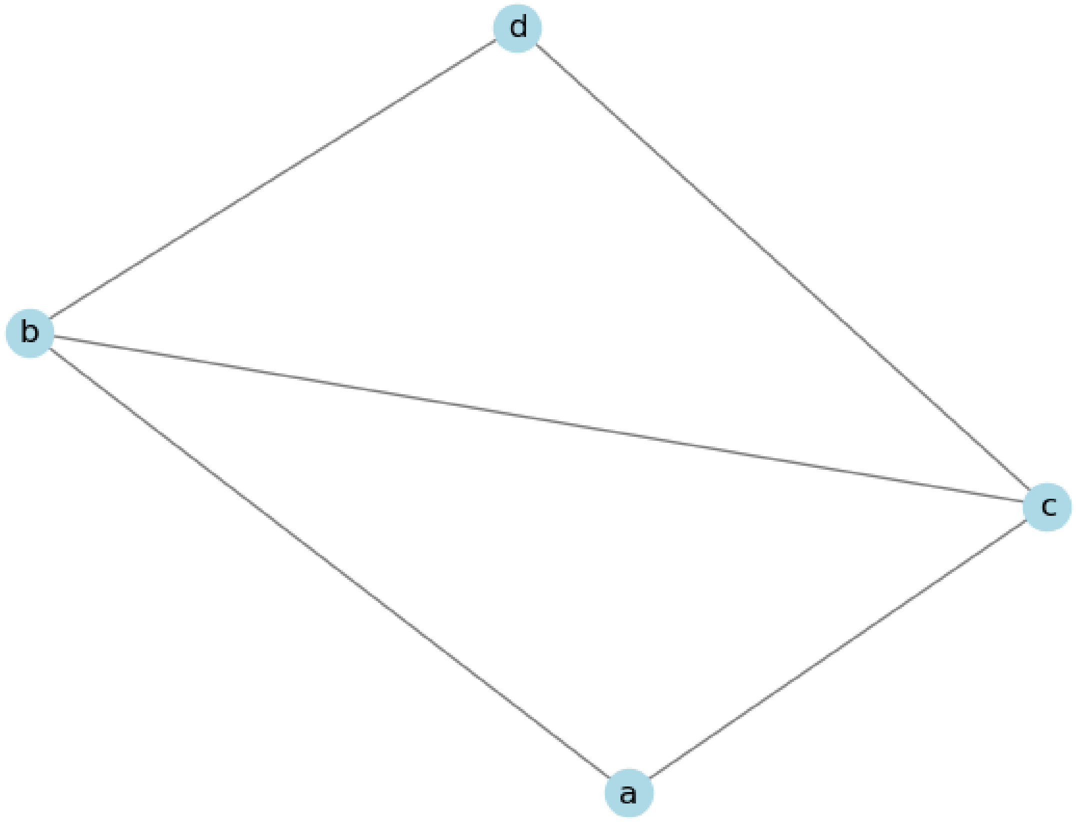
```
No. nodes: 3
No. edges: 2
```



In [26]:

```
# 최단경로 구하기1

import networkx as nx


G = nx.Graph()

G.add_edge('a','b',weight=3)

G.add_edge('b','c',weight=4)

G.add_edge('c','d',weight=3)

G.add_edge('d','b',weight=8)

G.add_edge('c','a',weight=10)

labels = nx.get_edge_attributes(G, 'weight')

print(labels)

nx.draw(G, with_labels=True, node_color='lightblue', edge_color='grey')
```

```
{('a', 'b'): 3, ('a', 'c'): 10, ('b', 'c'): 4, ('b', 'd'): 8, ('c', 'd'): 3}
```



In [27]:

```
# 그래프의 최단경로 구하기1

nx.shortest_path(G, 'a', 'd', weight='weight')
```

Out[27]:

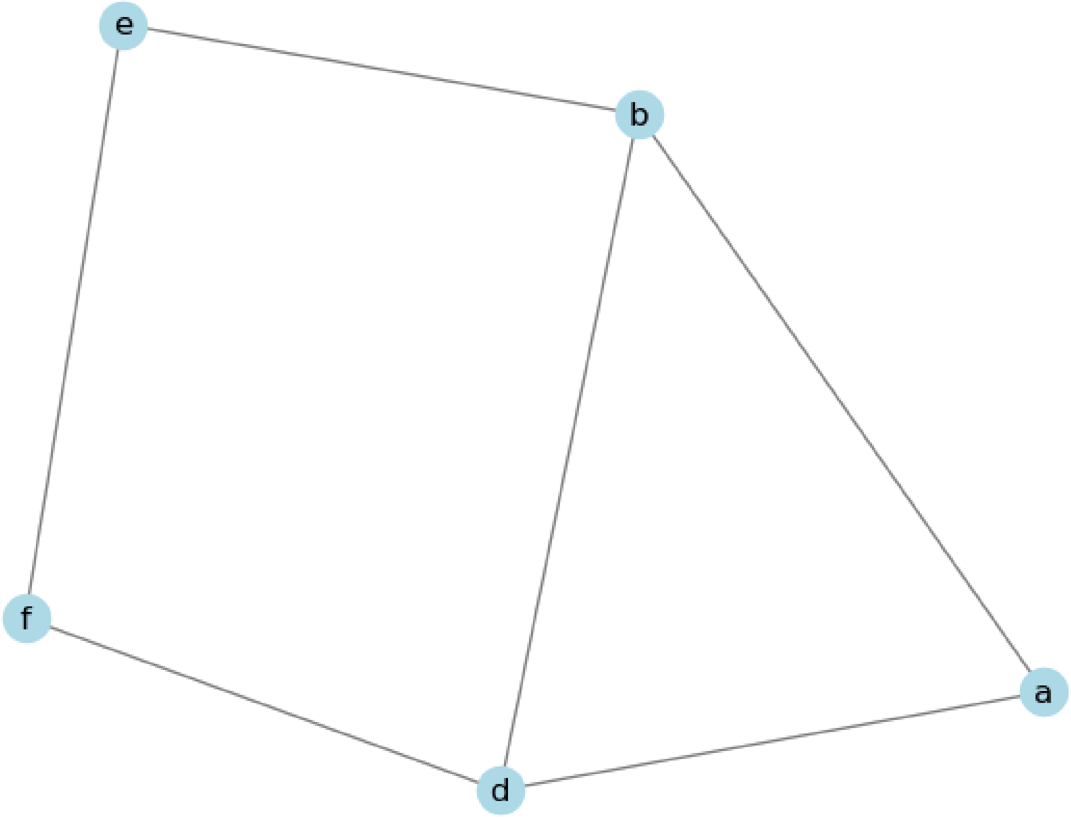
```
['a', 'b', 'c', 'd']
```

In [28]:

```
# 최단경로 구하기2

import networkx as nx


G = nx.Graph()

G.add_edge('a','b',weight=4)

G.add_edge('a','d',weight=8)

G.add_edge('b','d',weight=3)

G.add_edge('b','e',weight=10)

G.add_edge('f','d',weight=12)
```

```
G.add_edge('f','e',weight=7)

labels = nx.get_edge_attributes(G, 'weight')

print(labels)

nx.draw(G, with_labels=True, node_color='lightblue', edge_color='grey')
```

```
{('a', 'b'): 4, ('a', 'd'): 8, ('b', 'd'): 3, ('b', 'e'): 10, ('d', 'f'): 12,
('e', 'f'): 7}
```



In [29]:

```
# 그래프의 최단경로 구하기2

nx.shortest_path(G, 'a', 'f', weight='weight')
```

Out[29]:

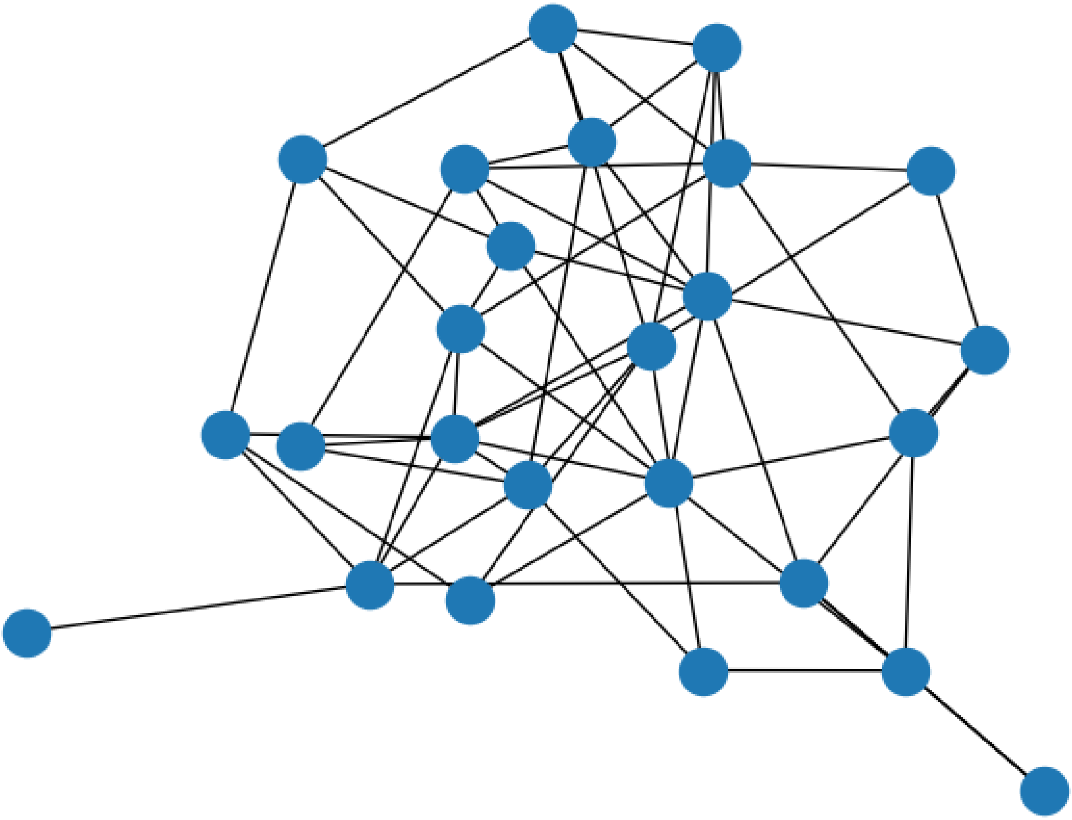
```
['a', 'b', 'd', 'f']
```

In [30]:

```
import networkx as nx

import matplotlib.pyplot as plt
```

```
G = nx.erdos_renyi_graph(25, 0.2)
```

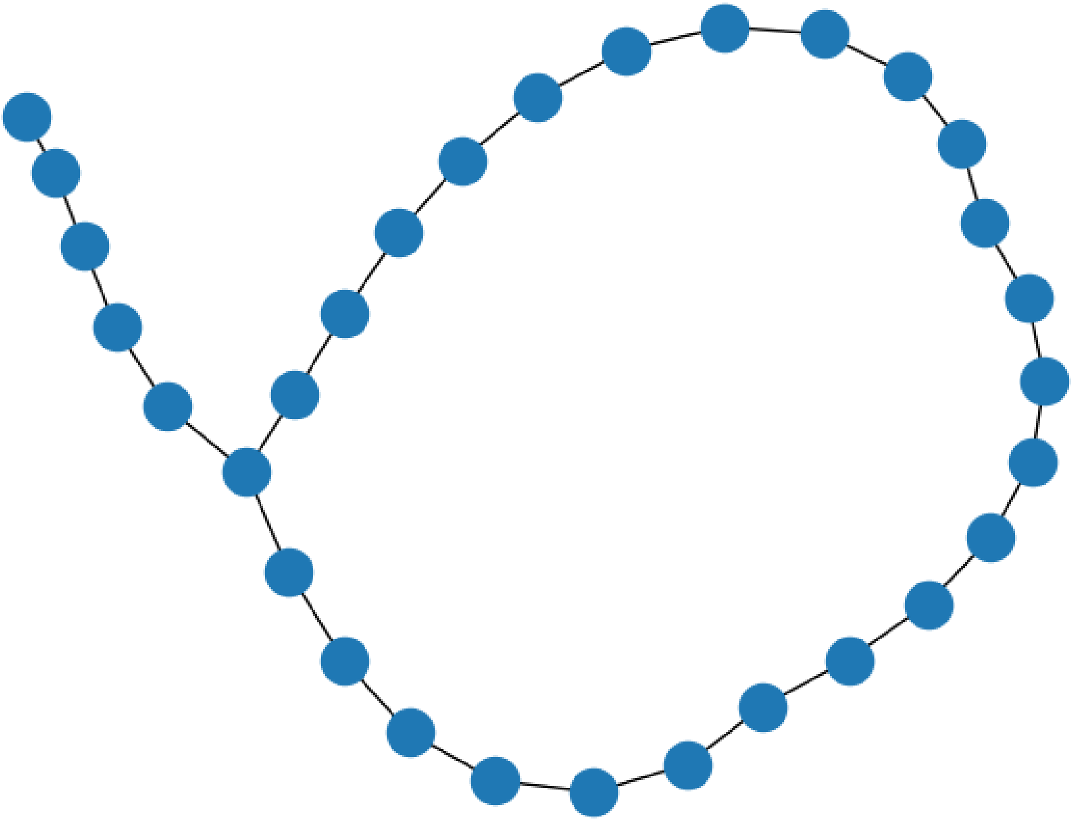
```
nx.draw(G)
```



In [31]:

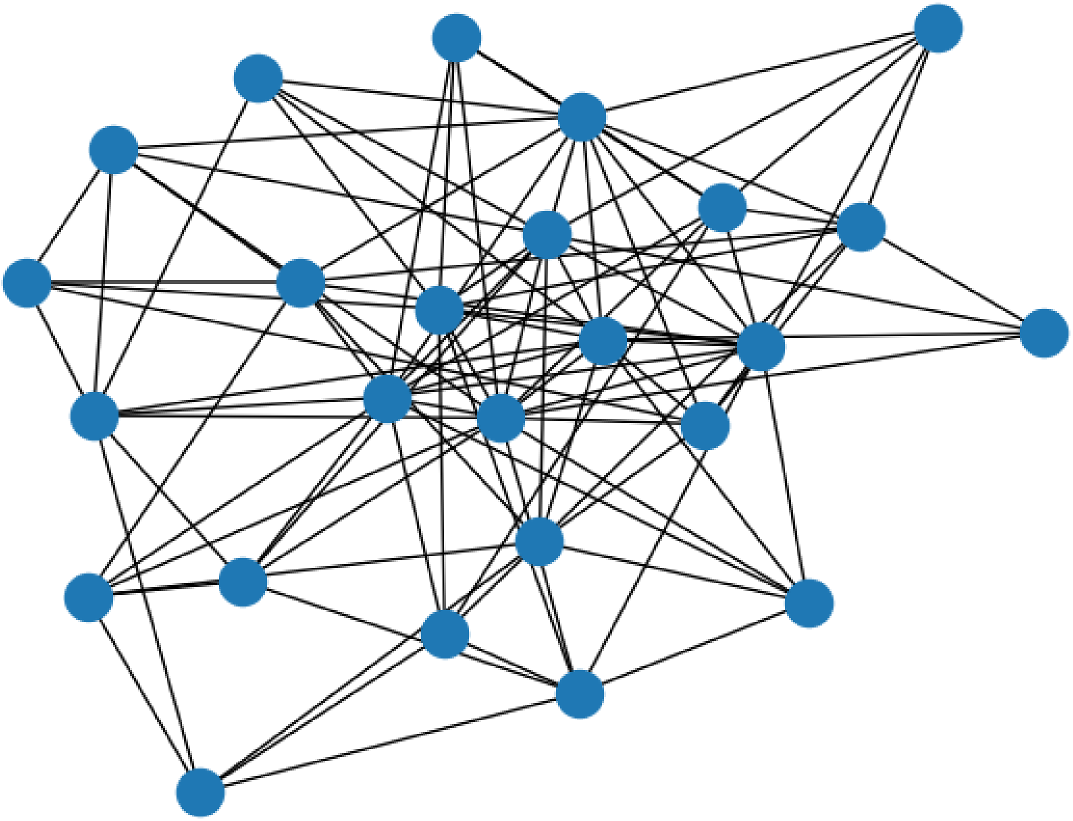
```
G = nx.watts_strogatz_graph(30, 3, 0.1)
```

```
nx.draw(G)
```



In [32]:

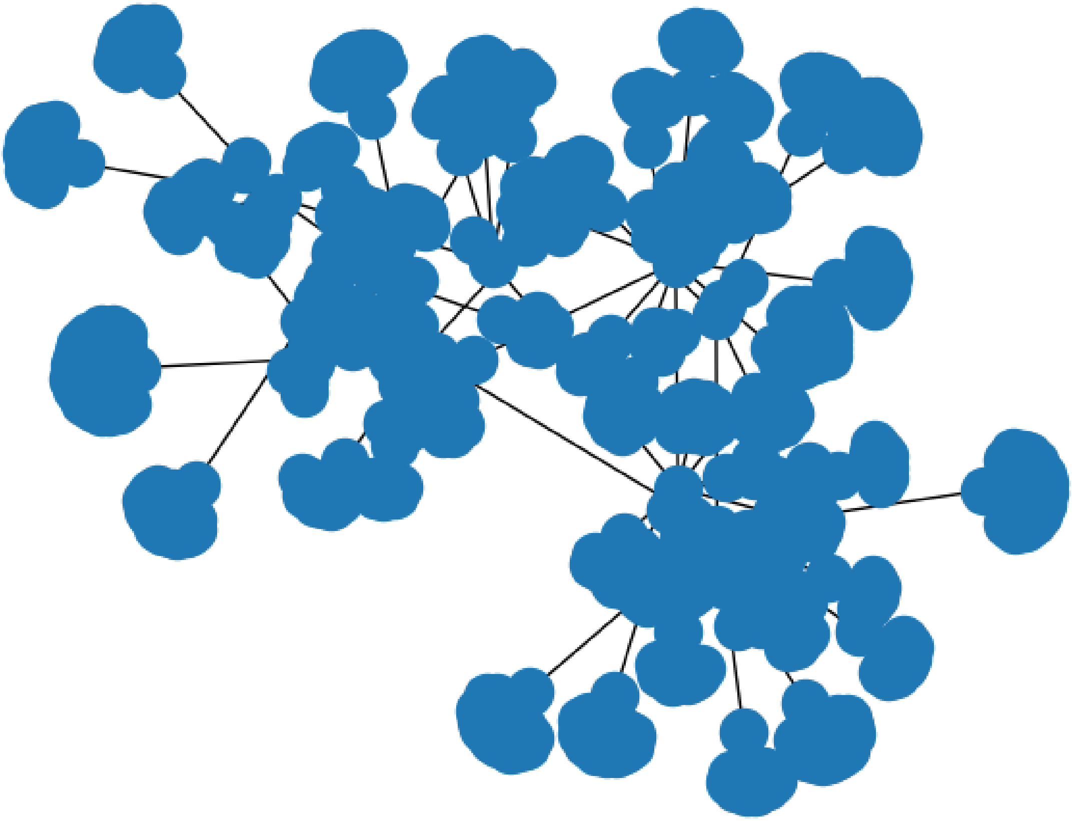
```
G = nx.barabasi_albert_graph(25, 5)

nx.draw(G)
```



In [33]:

```
G = nx.random_lobster(25, 0.9, 0.9)
```

```
nx.draw(G)
```



In [34]:

```
from scipy.stats import bernoulli



for i in range(6):

    print(bernoulli.rvs(p=0.33))
```

```
1
0
0
1
0
1
```

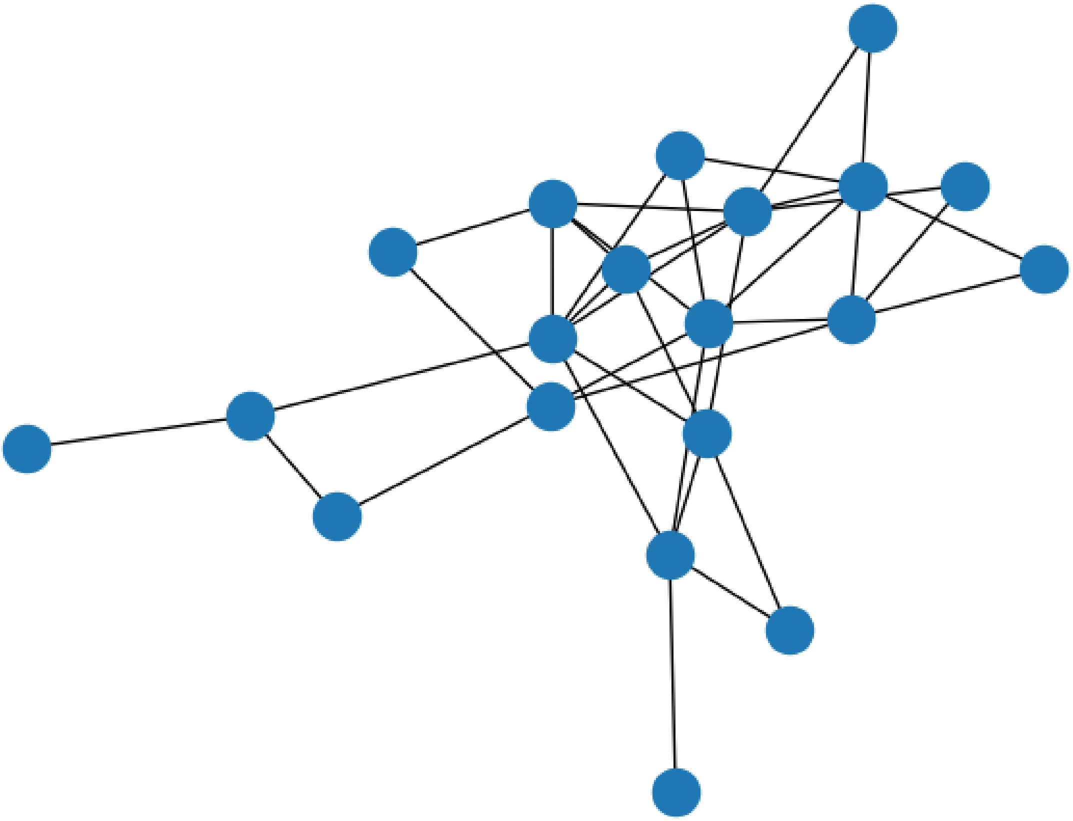
In [35]:

```
# erdosGraph() 함수 만들기

import numpy as np

import networkx as nx

import matplotlib.pyplot as plt
```

```
from scipy.stats import bernoulli


def erdosGraph(N, p):
    G = nx.Graph()
    G.add_nodes_from(range(N))
    listG = list(G.nodes())
    for i, node1 in enumerate(listG):
        for node2 in listG[i+1:]:
            if (bernoulli.rvs(p=p)):
                G.add_edge(node1, node2)
    return G


def pltGraph(G):
    plt.hist(list([d for n, d in G.degree()]), histtype='step')
```

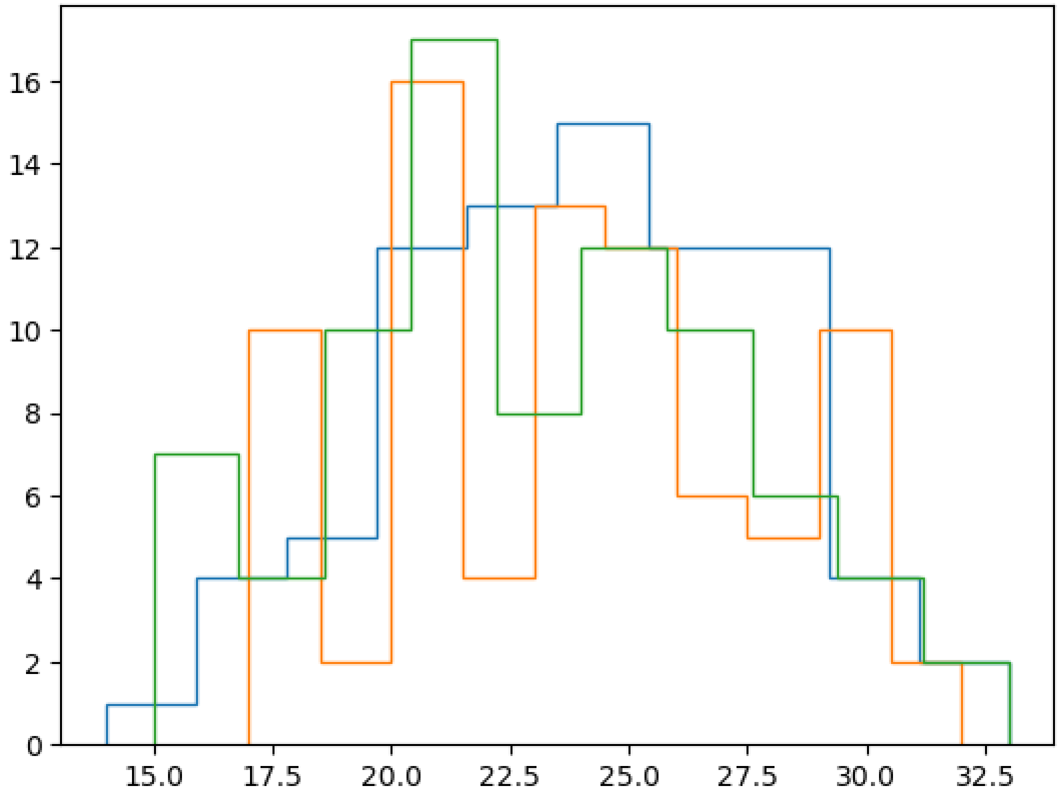
In [36]:

```
nx.draw(erdosGraph(20, 0.18))
```



In [37]:

```
G1 = erdosGraph(80, 0.3)

pltGraph(G1)

G2 = erdosGraph(80, 0.3)

pltGraph(G2)

G3 = erdosGraph(80, 0.3)

pltGraph(G3)
```



In [38]:

```
import matplotlib.pyplot as plt

import networkx as nx

from networkx.algorithms import bipartite



G = nx.davis_southern_women_graph()

women = G.graph["top"]

clubs = G.graph["bottom"]



print("Biadjacency matrix")

print(bipartite.biadjacency_matrix(G, women, clubs))



# project bipartite graph onto women nodes

W = bipartite.projected_graph(G, women)

print()

print("#Friends, Member")

for w in women:
```

```
print(f"{W.degree(w)} {w}")
```

```
Biadjacency matrix
  (0, 0)        1
  (0, 1)        1
  (0, 2)        1
  (0, 3)        1
  (0, 4)        1
  (0, 5)        1
  (0, 7)        1
  (0, 8)        1
  (1, 0)        1
  (1, 1)        1
  (1, 2)        1
  (1, 4)        1
  (1, 5)        1
  (1, 6)        1
  (1, 7)        1
  (2, 1)        1
  (2, 2)        1
  (2, 3)        1
  (2, 4)        1
  (2, 5)        1
  (2, 6)        1
  (2, 7)        1
  (2, 8)        1
  (3, 0)        1
  (3, 2)        1
  :     :
  (12, 7)       1
  (12, 8)       1
  (12, 9)       1
  (12, 11)      1
  (12, 12)      1
  (12, 13)      1
  (13, 5)       1
  (13, 6)       1
  (13, 8)       1
  (13, 9)       1
  (13, 10)      1
  (13, 11)      1
  (13, 12)      1
  (13, 13)      1
  (14, 6)       1
  (14, 7)       1
  (14, 9)       1
  (14, 10)      1
  (14, 11)      1
  (15, 7)       1
  (15, 8)       1
  (16, 8)       1
  (16, 10)      1
  (17, 8)       1
  (17, 10)      1

#Friends, Member
17 Evelyn Jefferson
15 Laura Mandeville
```

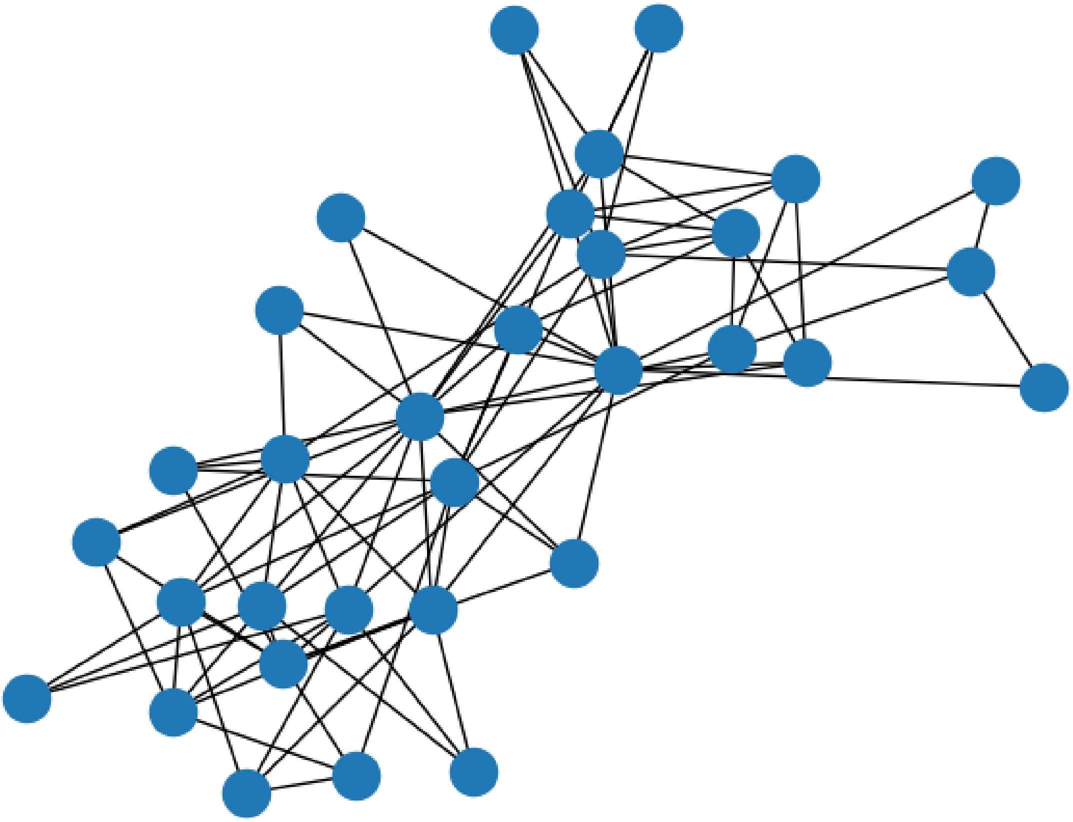
```
17 Theresa Anderson
15 Brenda Rogers
11 Charlotte McDowd
15 Frances Anderson
15 Eleanor Nye
16 Pearl Oglethorpe
17 Ruth DeSand
17 Verne Sanderson
16 Myra Liddel
16 Katherina Rogers
17 Sylvia Avondale
17 Nora Fayette
17 Helen Lloyd
16 Dorothy Murchison
12 Olivia Carleton
12 Flora Price
```

```
C:\Users\User\AppData\Local\Temp\ipykernel_28264\3234862610.py:10: FutureWarning:
biadjacency_matrix will return a scipy.sparse array instead of a matrix in
NetworkX 3.0
  print(bipartite.biadjacency_matrix(G, women, clubs))
```

## In [39]:

```
# project bipartite graph onto women nodes keeping number of co-occurrence

# the degree computed is weighted and counts the total number of shared contacts

W = bipartite.weighted_projected_graph(G, women)

print()

print("#Friend meetings, Member")

for w in women:

    print(f"{W.degree(w, weight='weight')} {w}")



pos = nx.spring_layout(G, seed=648)  # Seed layout for reproducible node positions

nx.draw(G, pos)

plt.show()
```

```
#Friend meetings, Member
50 Evelyn Jefferson
45 Laura Mandeville
57 Theresa Anderson
46 Brenda Rogers
24 Charlotte McDowd
32 Frances Anderson
36 Eleanor Nye
31 Pearl Oglethorpe
40 Ruth DeSand
38 Verne Sanderson
33 Myra Liddel
37 Katherina Rogers
46 Sylvia Avondale
43 Nora Fayette
34 Helen Lloyd
24 Dorothy Murchison
14 Olivia Carleton
14 Flora Price
```



In [ ]:

In [ ]: