Graph

1] Graph adjacency List representation in python :

```
def addEdge(adj,u,v):
    adj[u].append(v)
    adj[v].append(u)

def printGraph(adj):
    for u,l in enumerate(adj):
        print(u,l)

#main

v=4
adj=[[] for i in range(v)]
addEdge(adj,0,1)
addEdge(adj,0,2)
addEdge(adj,1,2)
addEdge(adj,1,3)

printGraph(adj)
```

OUTPUT :

```
0 [1, 2]

1 [0, 2, 3]

2 [0, 1]

3 [1]
```

2] Breadth First Search in Python :

```
from collections import deque

def addEdge(adj,u,v):
    adj[u].append(v)
    adj[v].append(u)

def BSF(adj,s):
    visited=[False]*len(adj)
```

```python
    q=deque()
    q.append(s)
    visited[s]=True

    while q:
        s=q.popleft()
        print(s,end=" ")

        for u in adj[s]:
            if visited[u]==False:
                q.append(u)
                visited[u]=True

def printGraph(adj):
    for u,l in enumerate(adj):
        print(u,l)

#main

v=4

adj=[[1,2],[0,2,3],[0,1,3,4],[1,2,4],[2,3]]

printGraph(adj)

s=0 #starting

print("\nBSF Path")
BSF(adj,s)
```

OUTPUT :

```
0 [1, 2]

1 [0, 2, 3]

2 [0, 1, 3, 4]

3 [1, 2, 4]

4 [2, 3]

BSF Path

0 1 2 3 4
```

3] BSF for Disconnected graph :

```
from collections import deque

def addEdge(adj,u,v):
    adj[u].append(v)
    adj[v].append(u)

def BSF(adj,s,visited):
    q=deque()
    q.append(s)
    visited[s]=True

    while q:
        s=q.popleft()
        print(s,end=" ")

        for u in adj[s]:
            if visited[u]==False:
                q.append(u)
                visited[u]=True

def BFSDis(adj):
    visited=[False]*len(adj)

    for u in range(len(adj)):
        if visited[u]==False:
            BSF(adj,u,visited)

def printGraph(adj):
    for u,l in enumerate(adj):
        print(u,l)

#main

v=7

adj=[[1,2],[0,3],[0,3],[1,2],[5,6],[4,6],[4,5]]

printGraph(adj)

print("\nBSF path")
BFSDis(adj)
```

OUTPUT :

```
0 [1, 2]
1 [0, 3]
2 [0, 3]
3 [1, 2]
4 [5, 6]
5 [4, 6]
6 [4, 5]

BSF path
0 1 2 3 4 5 6
```

4] Connected components in undirected graph :

```
from collections import deque

def addEdge(adj,u,v):
    adj[u].append(v)
    adj[v].append(u)

def BSF(adj,s,visited):
    q=deque()
    q.append(s)
    visited[s]=True

    while q:
        s=q.popleft()
        print(s,end=" ")
        for u in adj[s]:
            if visited[u]==False:
                q.append(u)
                visited[u]=t=True
    print()

def BSFDis(adj):
    visited=[False]*len(adj)
    res=0
    for u in range(len(adj)):
        if visited[u]==False:
            res+=1
            BSF(adj,u,visited)

    return res

def printGraph(adj):
    for u,l in enumerate(adj):
        print(u,l)


#main

v=8

adj=[[1,2],[0,2],[0,1],[4],[3],[6,7],[5],[5]]

printGraph(adj)
```

```
print("\nConnected Components")
print("no of connected components",BSFDis(adj))
```

OUTPUT :

```
0 [1, 2]

1 [0, 2]

2 [0, 1]

3 [4]

4 [3]

5 [6, 7]

6 [5]

7 [5]


Connected Components

0 1 2

3 4

5 6 7

no of connected components 3
```

5] Depth First Search :

```
def DFSRec(adj,s,visited):
    visited[s]=True

    print(s,end=" ")

    for u in adj[s]:
        if visited[u]==False:
            DFSRec(adj,u,visited)

def DFS(adj,s):
    visited=[False]*len(adj)
    DFSRec(adj,s,visited)

def printGraph(adj):
    for u,l in enumerate(adj):
        print(u,l)

adj=[[1,2],[0,3,4],[0,3],[1,2,4],[1,3]]

printGraph(adj)

DFS(adj,0)
```

OUTPUT :

```
0 [1, 2]

1 [0, 3, 4]

2 [0, 3]

3 [1, 2, 4]

4 [1, 3]

0 1 3 2 4
```

6] DFS for disconnected graph :

```
def DFSRec(adj,s,visited):
    visited[s]=True

    print(s,end=" ")

    for u in adj[s]:

        if visited[u]==False:
            DFSRec(adj,u,visited)

def DFS(adj):
    visited=[False]*len(adj)

    for u in range (len(adj)):
        if visited[u]==False:
            DFSRec(adj,u,visited)

def printGraph(adj):

    for u,l in enumerate(adj):
        print(u,l)

adj=[[1,2],[0,2],[0,1],[4],[3]]

printGraph(adj)

DFS(adj)
```

OUTPUT :

```
0 [1, 2]

1 [0, 2]

2 [0, 1]

3 [4]

4 [3]

0 1 2 3 4
```

7] connected component in undirected graph using DFS :

```
def DFSRec(adj,s,visited):
    visited[s]=True

    print(s,end=" ")

    for u in adj[s]:
        if visited[u]==False:
            DFSRec(adj,u,visited)

def DFS(adj):
    visited=[False]*len(adj)
    res=0
    for u in range(len(adj)):
        if visited[u]==False:
            res+=1
            DFSRec(adj,u,visited)
            print()
    return  res

def printGraph(adj):
    for u,l in enumerate(adj):
        print(u,l)

adj=[[1,2],[0,2],[0,1],[4],[3]]

printGraph(adj)

print("conneted component")
print("no of connected component ",DFS(adj))
```

OUTPUT :

```
0 [1, 2]

1 [0, 2]

2 [0, 1]

3 [4]

4 [3]
```

```
conneted component

0 1 2

3 4

no of connected component  2
```