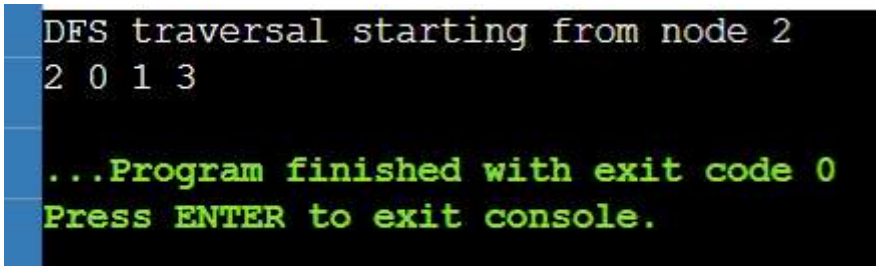
ASSIGNMENT-01

DFS for directed graph:

```
from collections import defaultdict

class Graph:
    def __init__(self):
        self.graph = defaultdict(list)

    def add_edge(self, u, v):
        self.graph[u].append(v)

    def dfs(self, start_node, visited):
        visited.add(start_node)
        print(start_node, end=" ")

        for neighbor in self.graph[start_node]:
            if neighbor not in visited:
                self.dfs(neighbor, visited)

# Create a directed graph
graph = Graph()
graph.add_edge(0, 1)
graph.add_edge(0, 2)
graph.add_edge(1, 2)
graph.add_edge(2, 0)
graph.add_edge(2, 3)
graph.add_edge(3, 3)

start_node = 2
visited = set()
print("DFS traversal starting from node", start_node)
graph.dfs(start_node, visited)
```

Output:



DFS for Undirected graph:

```
from collections import defaultdict

class Graph:
    def __init__(self):
        self.graph = defaultdict(list)

    def add_edge(self, u, v):
        self.graph[u].append(v)
        self.graph[v].append(u)

    def dfs(self, start_node, visited):
        visited.add(start_node)
        print(start_node, end=" ")

        for neighbor in self.graph[start_node]:
            if neighbor not in visited:
                self.dfs(neighbor, visited)

# Create an undirected graph
graph = Graph()
graph.add_edge(0, 1)
graph.add_edge(0, 2)
graph.add_edge(1, 2)
graph.add_edge(1, 3)
graph.add_edge(2, 4)
graph.add_edge(3, 4)
graph.add_edge(4, 5)
```

```
start_node = 0
visited = set()
print("DFS traversal starting from node", start_node)
graph.dfs(start_node, visited)
```

Output: