

**EXPERIMENT NO: 4**  
**DATE: 26/02/2021**

**A N V SREEVISHNU**  
**RA1811003010333**

## **IMPLEMENTATION OF BFS & DFS**

**Aim:** To implement BFS & DFS in AI using Python.

### **Procedure/Algorithm:**

- First, Create a graph with number of nodes of your choice.
- Traverse through the graph with moving towards left.
- If a node is not visited pop out into the queue and print.
- If visited the node moves on and traverses to its neighbour depending on bfs or dfs it would traverse accordingly.
- Once all the nodes have been traversed, the code ends printing out all the nodes.

### **Code:**

BFS:

```
graph = {
```

```
'A' : ['B'],
```

```
'B' : ['E', 'C'],
```

```
'C' : ['D'],
```

```
'D' : [],
```

```
'E' : ['F'],
```

```
'F' : []
```

```
}
```

```
visited = []
```

```
queue = []
```

```
def bfs(visited, graph, node):
```

```
    visited.append(node)
```

```
    queue.append(node)
```

```
    while queue:
```

```
        s = queue.pop(0)
```

```
        print (s, end = " ")
```

```
for neighbour in graph[s]:
```

```
    if neighbour not in visited:
```

```
        visited.append(neighbour)
```

```
        queue.append(neighbour)
```

```
bfs(visited, graph, 'A')
```

DFS:

```
graph = {
```

```
    'A' : ['B'],
```

```
    'B' : ['E', 'C'],
```

```
    'C' : ['D'],
```

```
    'D' : [],
```

```
'E' : ['F'],
```

```
'F' : []
```

```
}
```

```
visited = set()
```

```
def dfs(visited, graph, node):
```

```
    if node not in visited:
```

```
        print (node , end=' ')
```

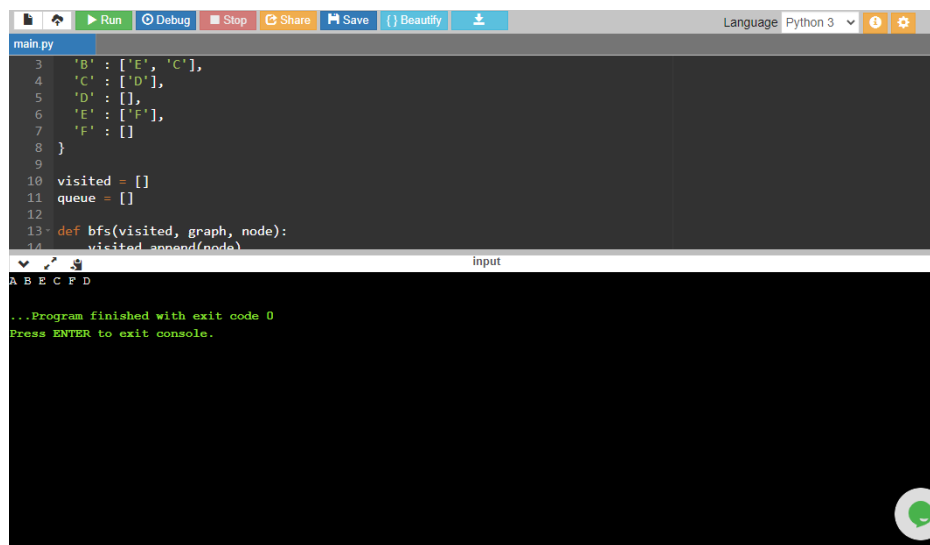
```
        visited.add(node)
```

```
        for neighbour in graph[node]:
```

```
            dfs(visited, graph, neighbour)
```

dfs(visited, graph, 'A')

## Output:



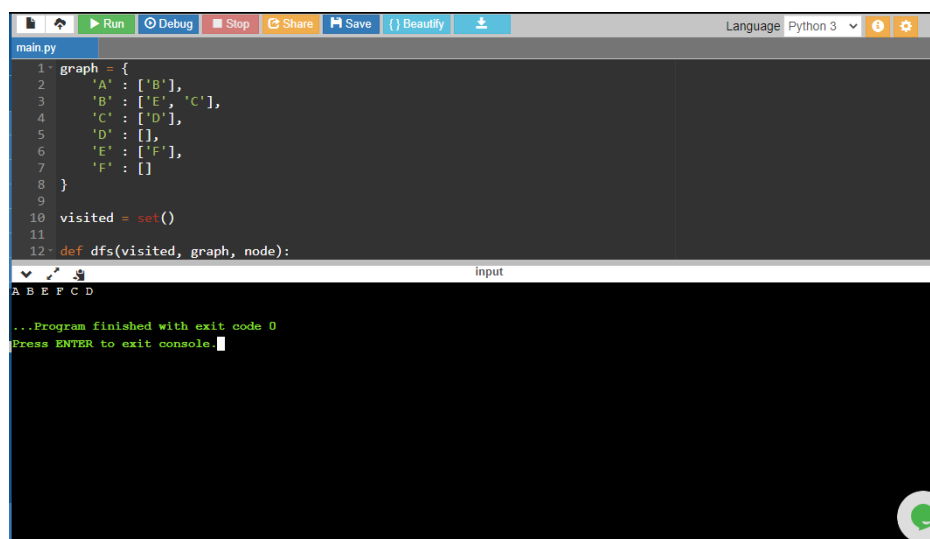
The screenshot shows a Python IDE with a file named 'main.py'. The code defines a graph with nodes 'A' through 'F' and their neighbors. A BFS function is implemented, and the graph is traversed starting from node 'A'. The output in the console shows the nodes visited in the order: A, B, E, C, F, D.

```
main.py
3  'B' : ['E', 'C'],
4  'C' : ['D'],
5  'D' : [],
6  'E' : ['F'],
7  'F' : []
8  }
9
10 visited = []
11 queue = []
12
13 def bfs(visited, graph, node):
14     visited.append(node)
```

input

A B E C F D

...Program finished with exit code 0  
Press ENTER to exit console.



The screenshot shows a Python IDE with a file named 'main.py'. The code defines a graph with nodes 'A' through 'F' and their neighbors. A DFS function is implemented, and the graph is traversed starting from node 'A'. The output in the console shows the nodes visited in the order: A, B, E, F, C, D.

```
main.py
1  graph = {
2      'A' : ['B'],
3      'B' : ['E', 'C'],
4      'C' : ['D'],
5      'D' : [],
6      'E' : ['F'],
7      'F' : []
8  }
9
10 visited = set()
11
12 def dfs(visited, graph, node):
```

input

A B E F C D

...Program finished with exit code 0  
Press ENTER to exit console.

**Result:** Thus, the implementation of BFS & DFS in AI using Python has been successfully done.