**Artificial Intelligence Lab**

**LAB 4 – DFS and BFS**

**Aditya Kumar**

**RA1911030010100**

**O2 Section**

**Date – 4/2/22**

**Problem Statement :**

Implementation and analysis DFS and BFS for an application.

**Algorithm:**

**For DFS:**

1. We will start by putting any one of the graph's vertex on top of the stack.

2. After that take the top item of the stack and add it to the visited list of the vertex.

3. Next, create a list of that adjacent node of the vertex. Add the ones which aren't in the visited list of vertexes to the top of the stack.

4. Lastly, keep repeating steps 2 and 3 until the stack is empty.

**For BFS:**

1. Start by putting any one of the graph’s vertices at the back of the queue.

2. Now take the front item of the queue and add it to the visited list.

3. Create a list of that vertex's adjacent nodes. Add those which are not within the visited list to the rear of the queue.

4. Keep continuing steps two and three till the queue is empty.

**CODE:**

**For DFS:**

graph = {

'5' : ['3','7'],

'3' : ['2', '4'],

'7' : ['8'],

'2' : [],

'4' : ['8'],

'8' : []

}

visited = set()

def dfs(visited, graph, node):

if node not in visited:

print (node)

visited.add(node)

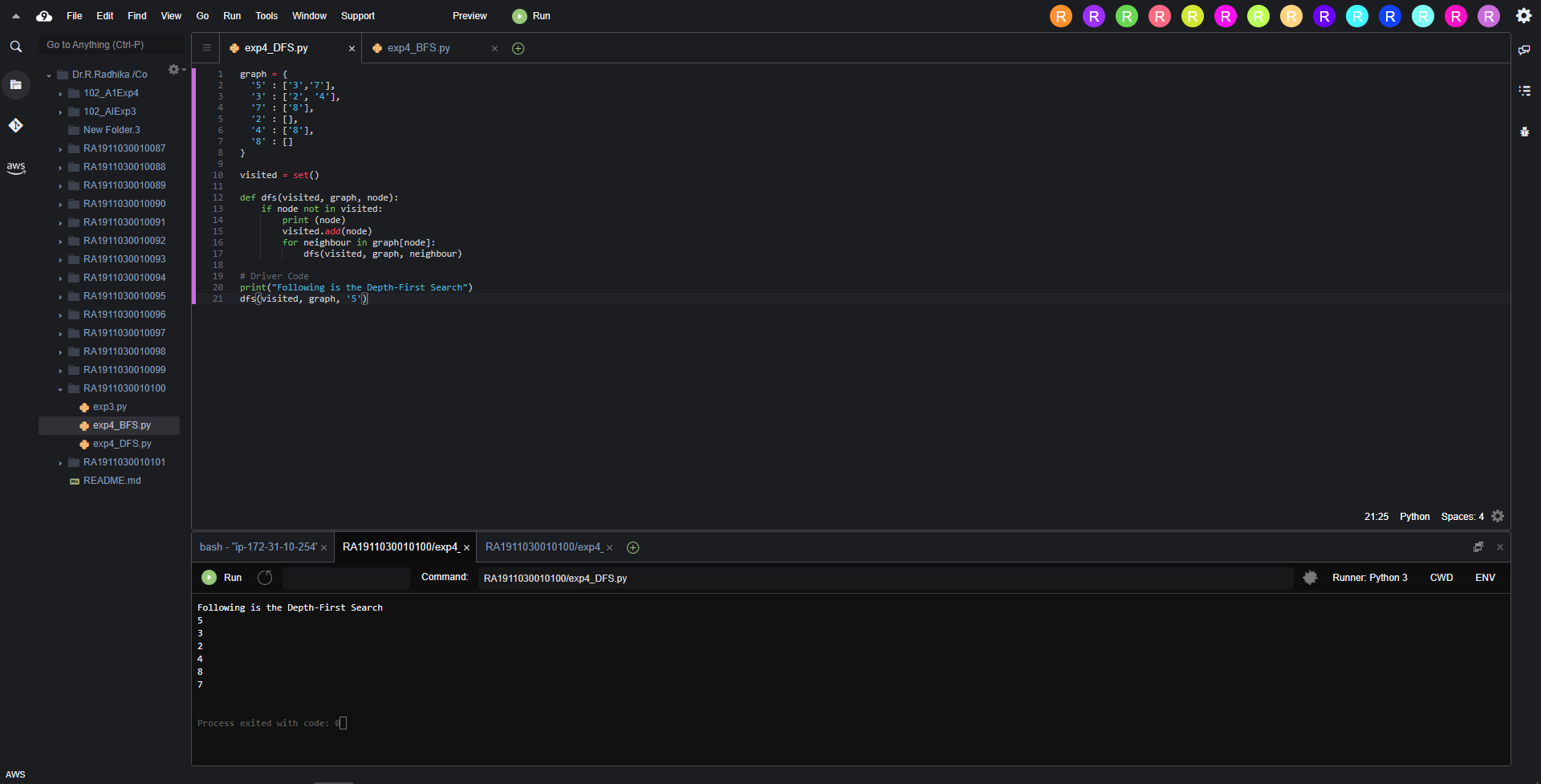
for neighbour in graph[node]:

dfs(visited, graph, neighbour)

print("Following is the Depth-First Search")

dfs(visited, graph, '5')

**Output:**



**For BFS:**

graph = {

'5' : ['3','7'],

'3' : ['2', '4'],

'7' : ['8'],

'2' : [],

'4' : ['8'],

'8' : []

}

visited = []

queue = []

def bfs(visited, graph, node):

visited.append(node)

queue.append(node)

while queue:

m = queue.pop(0)

print (m, end = " ")

for neighbour in graph[m]:

if neighbour not in visited:

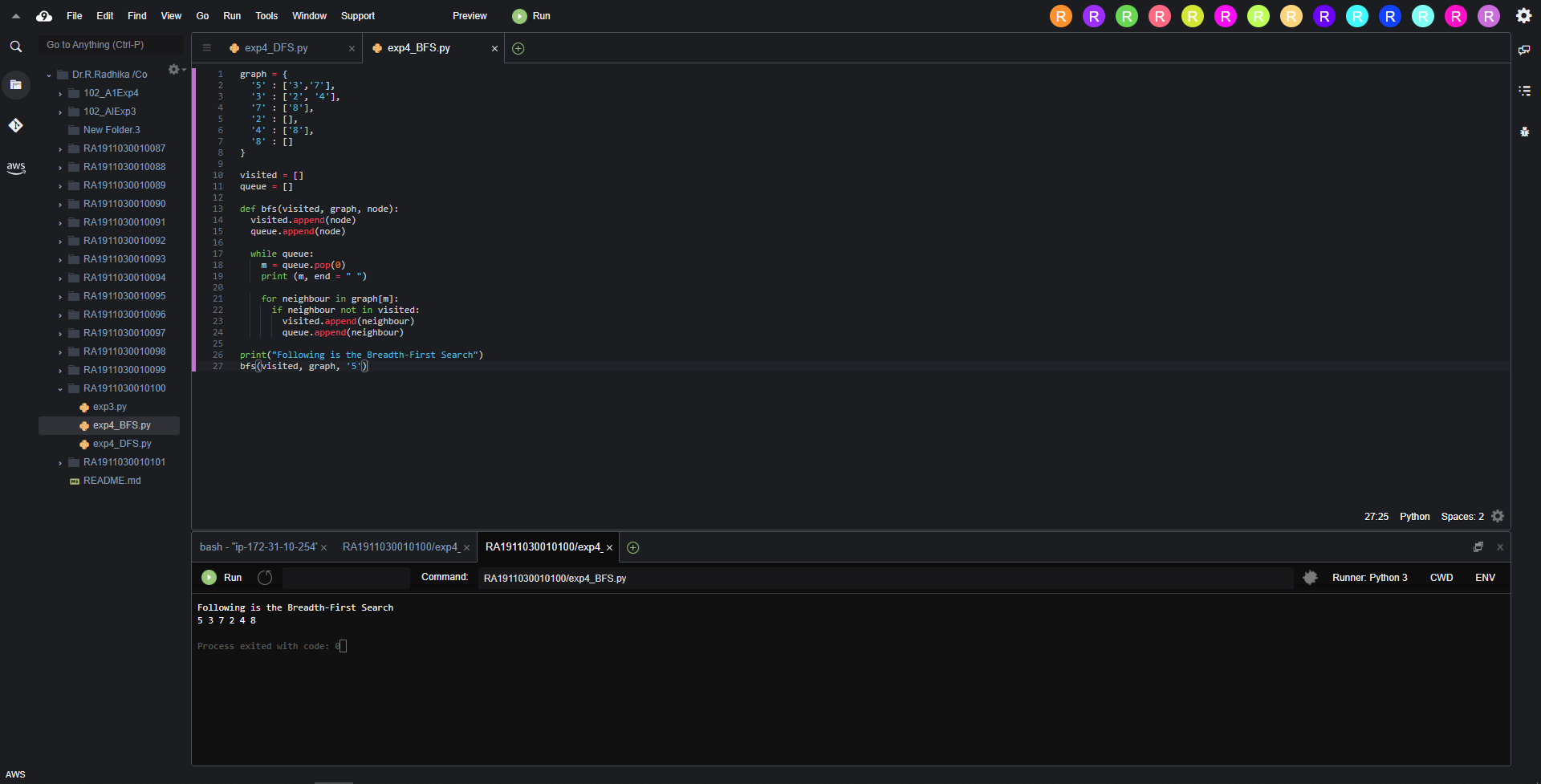
visited.append(neighbour)

queue.append(neighbour)

print("Following is the Breadth-First Search")

bfs(visited, graph, '5')

**Output:**



Result:

Hence we successfully implemented and executed DFS and BFS for an application.