

**ABUBAKAR ASIF**

***#Implementation of Depth-First Search (DFS)***

**#Implementation of Breadth-First Search**

**#abubakarasif3111@gmail.com**

**#https://github.com/Abubakar3111**

**#https://www.linkedin.com/in/abubakar-asif-b3b94021a/**

**Artificial Intelligence**

**#Implementation of Breadth-First Search**

**from search import \***

**graph={'A':{'B','C'},'B':{'D','E'},'C':{'F','G'},'D':{},'E':{},'F':{},'G':{}}**

**class Node():**

**def \_init\_(self,state,parent=None,cost=0,heuristic=0):**

**self.state=state**

**self.parent=parent**

**self.cost=cost**

**self.heuristic=heuristic**

**def \_repr\_(self):**

**return repr(self.state)**

**def goal\_test(state):**

**if state==goal:**

**return True**

**else:**

**return False**

**def sucessors(state):**

**return graph[state]**

**def node\_to\_path(node):**

**path=[node.state]**

**while node.parent != None:**

**node=node.parent**

**path.append(node.state)**

**path.reverse()**

**#print(path)**

**return path**

**def bfs(initial):**

**frontier=Queue()**

**inode=Node(initial)**

**frontier.push(inode)**

**explored={initial}**

**while frontier.empty:**

**print(frontier)**

**print('hello')**

**current\_node=frontier.pop()**

**current\_state=current\_node.state**

**if goal\_test(current\_state):**

**print(node\_to\_path(current\_node))**

**return current\_node**

**for child in sucessors(current\_state):**

**if child not in explored:**

**frontier.push(Node(child,current\_node))**

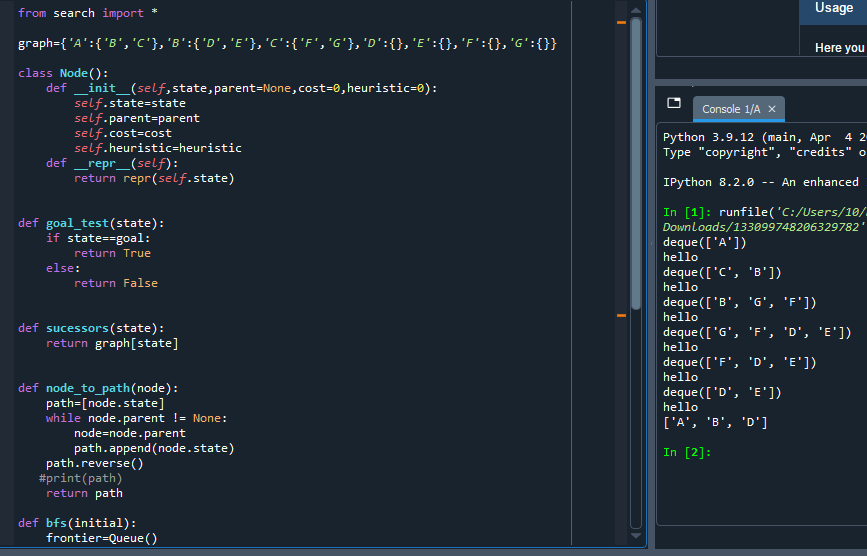
**explored.add(child)**

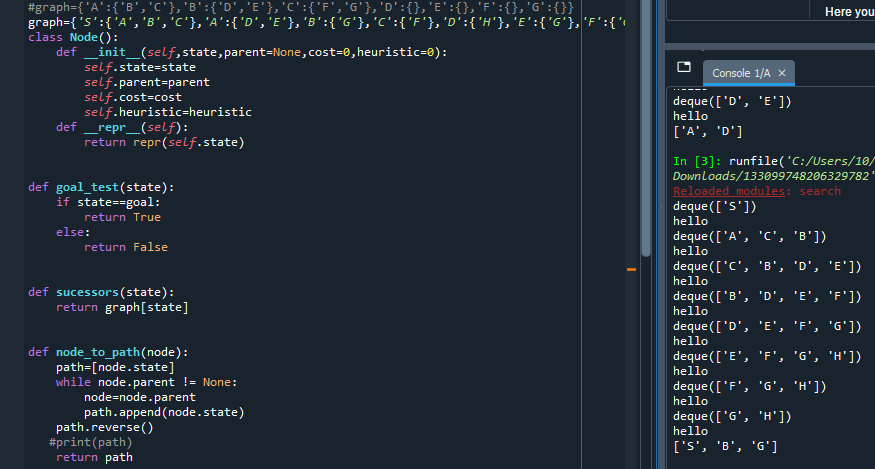
**return None**

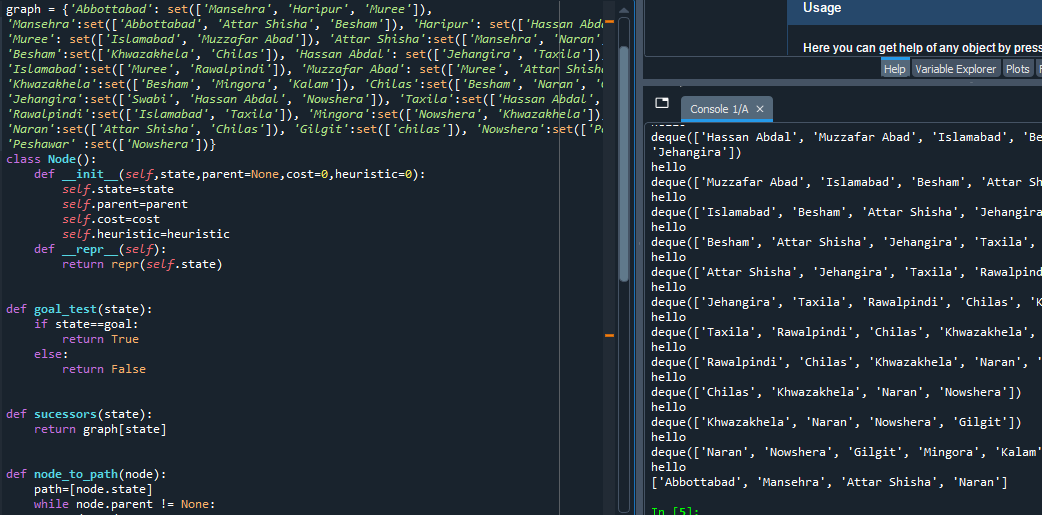
**start='A'**

**goal='D'**

**bfs(start)**

****

****

****

***#Implementation of Depth-First Search (DFS):***

**from search import \***

**#graph={'A':{'B','C'},'B':{'D','E'},'C':{'F','G'},'D':{},'E':{},'F':{},'G':{}}**

**graph={'S':{'A','B','C'},'A':{'D','E'},'B':{'G'},'C':{'F'},'D':{'H'},'E':{'G'},'F':{'G'}}**

**class Node():**

**def \_init\_(self,state,parent=None,cost=0,heuristic=0):**

**self.state=state**

**self.parent=parent**

**self.cost=cost**

**self.heuristic=heuristic**

**def \_repr\_(self):**

**return repr(self.state)**

**def goal\_test(state):**

**if state==goal:**

**return True**

**else:**

**return False**

**def sucessors(state):**

**return graph[state]**

**def node\_to\_path(node):**

**path=[node.state]**

**while node.parent != None:**

**node=node.parent**

**path.append(node.state)**

**path.reverse()**

**#print(path)**

**return path**

**def dfs(initial):**

**frontier=Stack()**

**inode=Node(initial)**

**frontier.push(inode)**

**explored={initial}**

**while frontier.empty:**

**print(frontier)**

**print('hello')**

**current\_node=frontier.pop()**

**current\_state=current\_node.state**

**if goal\_test(current\_state):**

**print(node\_to\_path(current\_node))**

**return current\_node**

**for child in sucessors(current\_state):**

**if child not in explored:**

**frontier.push(Node(child,current\_node))**

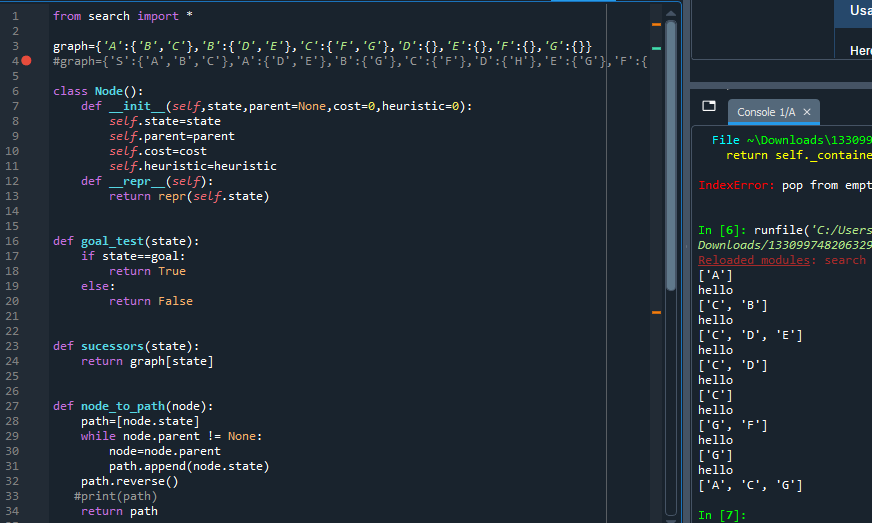
**explored.add(child)**

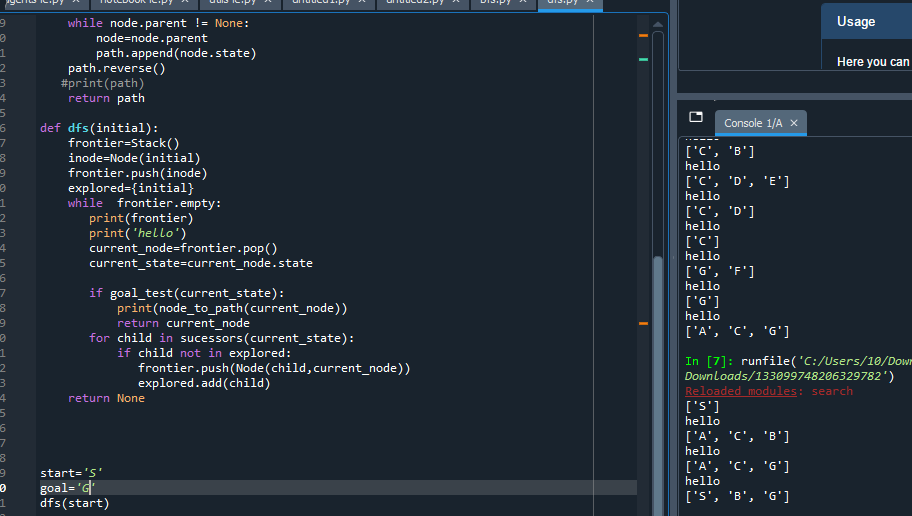
**return None**

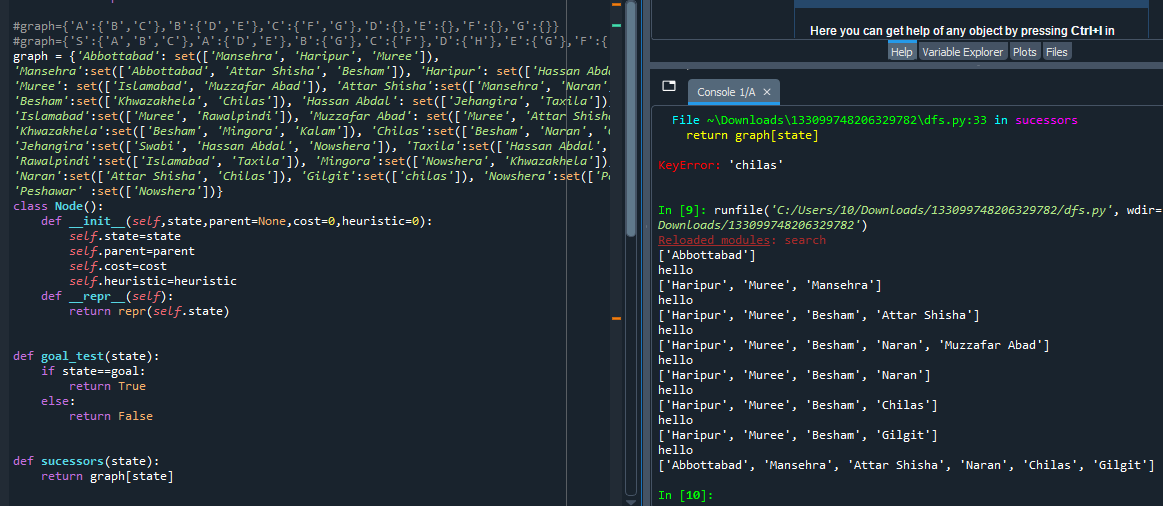
**start='S'**

**goal='G'**

**dfs(start)**

****

****

****