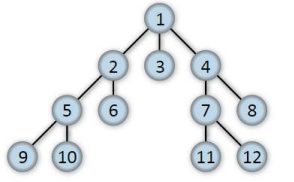
**Qno.1)**

→ **Objective:** To write python program implementing breadth first search

→ **Task name:**  Construct the given tree, write a breadth first search function ***def  bfs(graph, start\_node, goal\_node*)** which should return goal node if the  data is found in the tree else return None.

→ **Code:**

**def bfs\_search(graph, starting\_node, goal\_node):**

**visited = set()**

**queue = []**

**if starting\_node == goal\_node:**

**return [starting\_node]**

**visited.add(starting\_node)**

**queue.append(starting\_node)**

**while queue:**

**path = queue.pop(0)**

**print(path, end='->' if path != f'{goal\_node}' else '')**

**if path == goal\_node:**

**return path**

**for neighbour in graph[path]:**

**if neighbour not in visited:**

**visited.add(neighbour)**

**queue.append(neighbour)**

**print('None')**

**return None**

**graph = {**

**'1': ['2', '3', '4'],**

**'2': ['5', '6'],**

**'3': [],**

**'4': ['7', '8'],**

**'5': ['9', '10'],**

**'6': [],**

**'7': ['11', '12'],**

**'8': [],**

**'9': [],**

**'10': [],**

**'11': [],**

**'12': []**

**}**

**starting\_node = '1'**

**goal\_node = '6'**

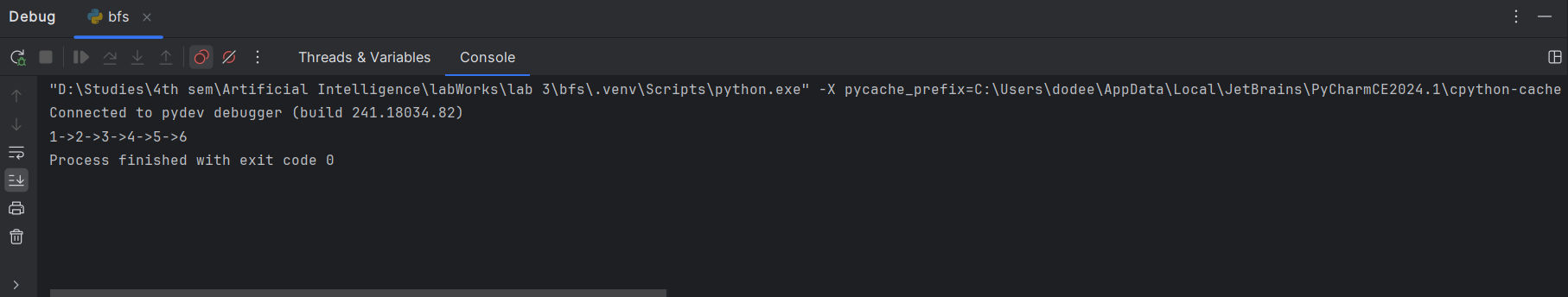
**bfs\_search(graph, starting\_node, goal\_node)**

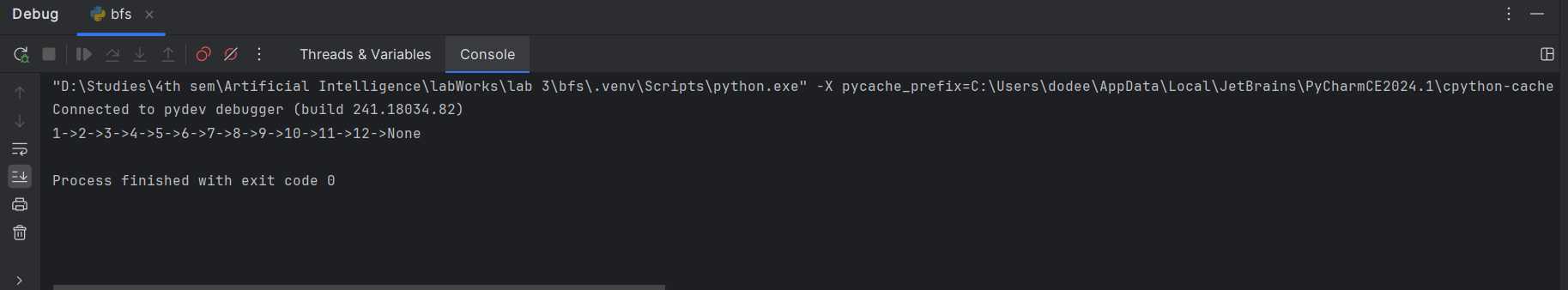
**starting\_node = '1'**

**goal\_node = '6'**

**bfs\_search(graph, starting\_node, goal\_node)**

→ **Output:**

1. When goal is 6
2. When goal is not in the node



**Qno.2)**

→ **Objective:** To write python program implementing depth first search

→ **Task name:** Construct the tree as in the previous question, write a depth first search function def dfs(graph, start\_node, goal\_node) which should return goal node if the data is found in the tree else return None.

→ **Code:**

**def bfs\_search(graph, starting\_node, goal\_node):**

**visited = set()**

**stack = []**

**if starting\_node == goal\_node:**

**return starting\_node**

**visited.add(starting\_node)**

**stack.append(starting\_node)**

**while stack:**

**path = stack.pop()**

**print(path, end='->' if path != f'{goal\_node}' else '')**

**if path == goal\_node:**

**return path**

**for neighbour in reversed(graph[path]):**

**if neighbour not in visited:**

**visited.add(neighbour)**

**stack.append(neighbour)**

**print('None')**

**return None**

**graph = {**

**'1': ['2', '3', '4'],**

**'2': ['5', '6'],**

**'3': [],**

**'4': ['7', '8'],**

**'5': ['9', '10'],**

**'6': [],**

**'7': ['11', '12'],**

**'8': [],**

**'9': [],**

**'10': [],**

**'11': [],**

**'12': []**

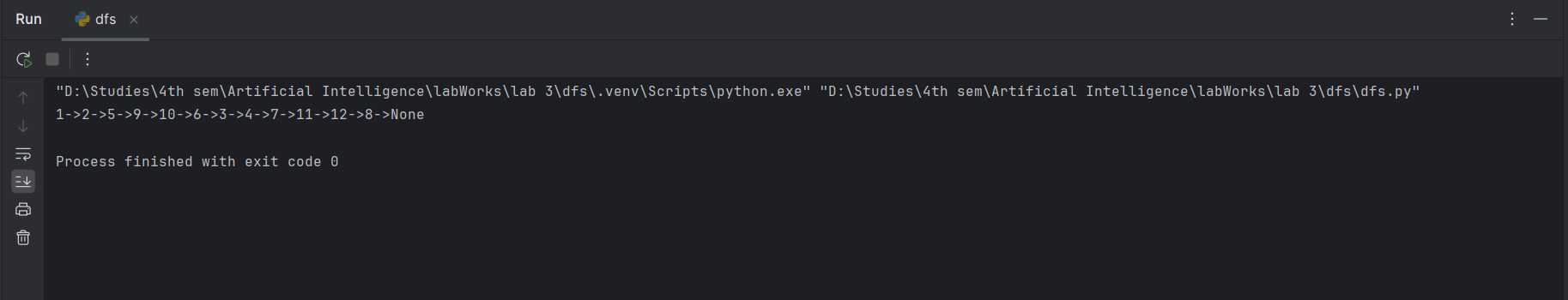
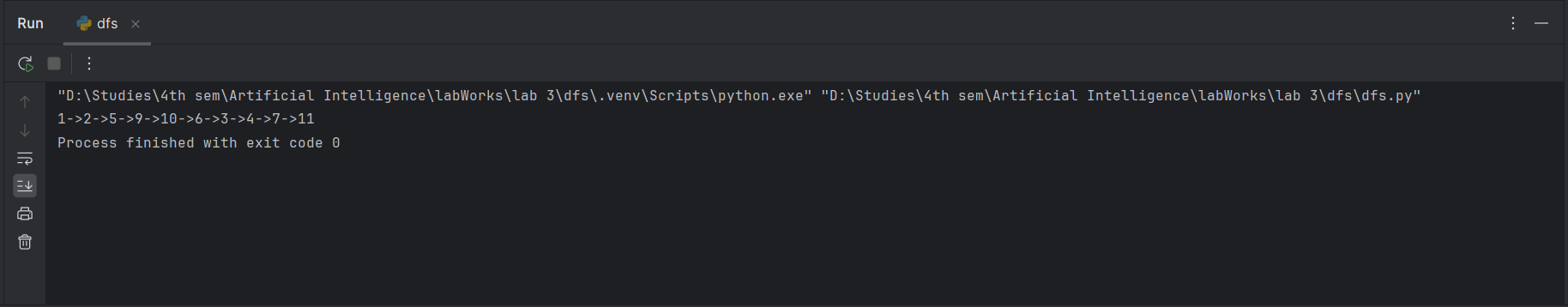
**}**

**starting\_node = '1'**

**goal\_node = '11'**

**bfs\_search(graph, starting\_node, goal\_node)**

→ **Output :**

1. **When goal is not in node**
2. **When goal is 11**