그래프의 탐색

그래프의 탐색 ?

탐색 : 모든 정점과 간선을 검사함으로써 그래프를 탐험하는 체계적인 절차

- 수도권 전철망의 모든 역(정점)의 위치를 출력
- 항공사의 모든 항공편(간선)에 대한 노선 정보를 수집

- DFS(깊이 우선 탐색, Depth First Search)
- BFS(너비 우선 탐색, Breath First Search)

DFS 깊이 우선 탐색

‘스택‘ 자료구조를 사용한 그래프 탐색 알고리즘

루트 노드에서 시작하여 다른 분기(Branch)로 넘어가기 전, 현재 탐색중인 분기를 완벽하게 (깊게) 탐색하는 방식

1. 루트 노드를 스택에 넣고 방문처리 한다.
2. 스택 최상단 노드의 인접 노드 중 방문하지 않은 노드 하나를 스택에 넣고 방문처리 한다. 만약 인접 노드를 모두 방문한 경우 스택을 pop한다.
3. 2단계를 더 이상 수행할 수 없을 때 까지 반복한다.

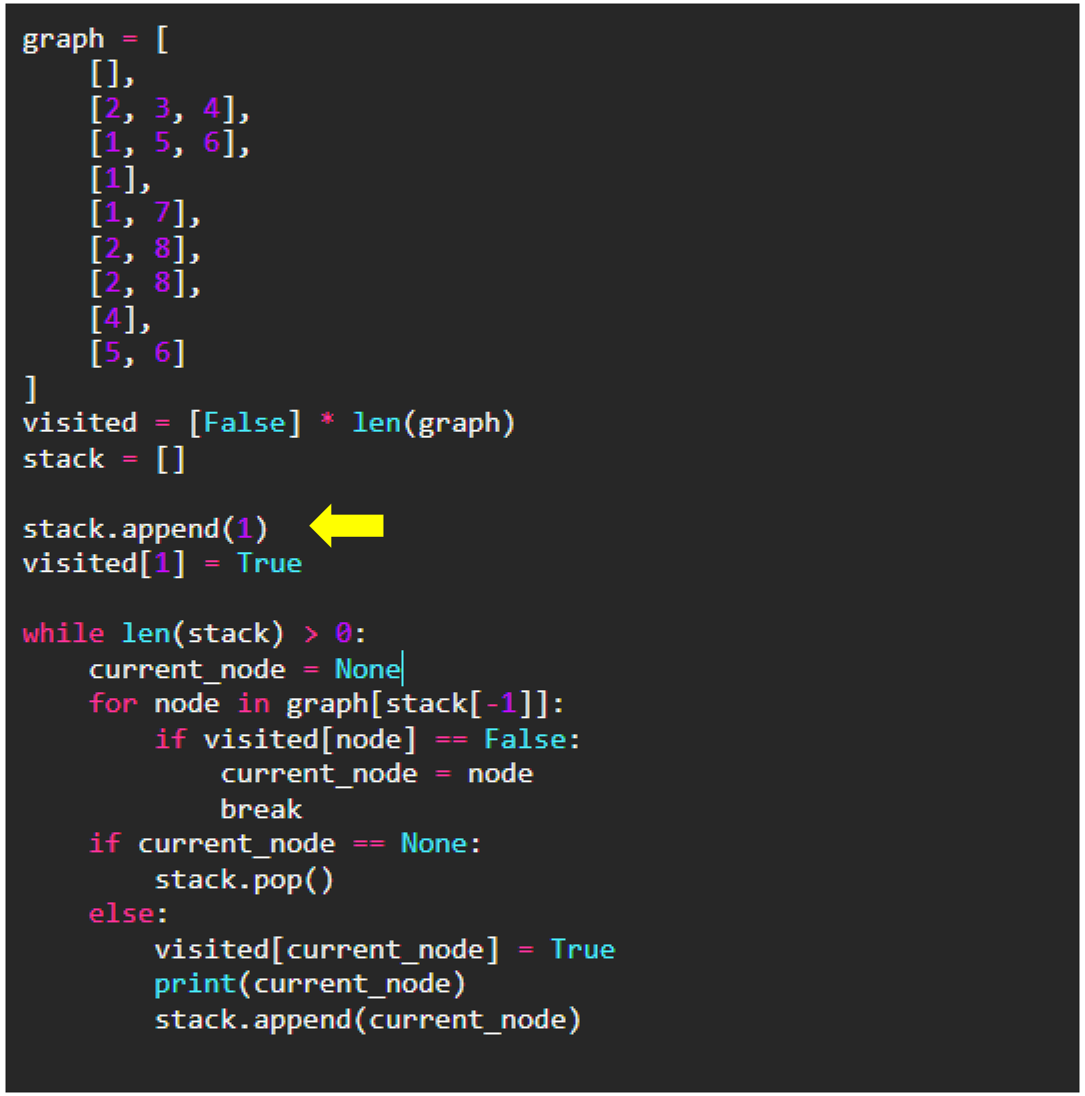
DFS 깊이 우선 탐색

장점

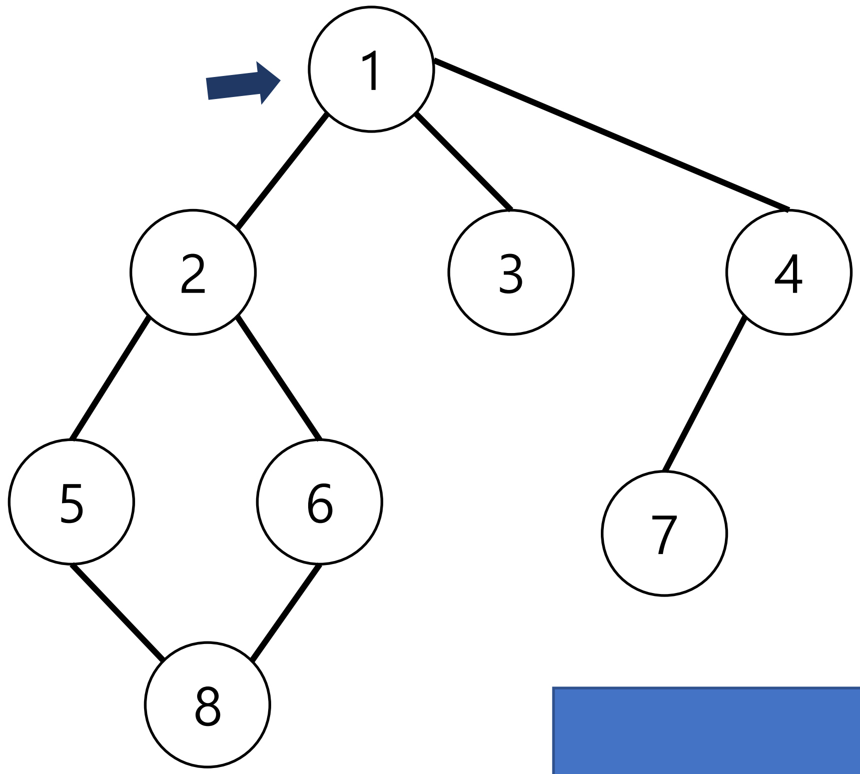
- 현재 경로상의 노드들만 기억하면 되므로, 저장 공간의 수요가 비교적 적음
- 구현이 너비 우선 탐색(BFS) 보다 간단함

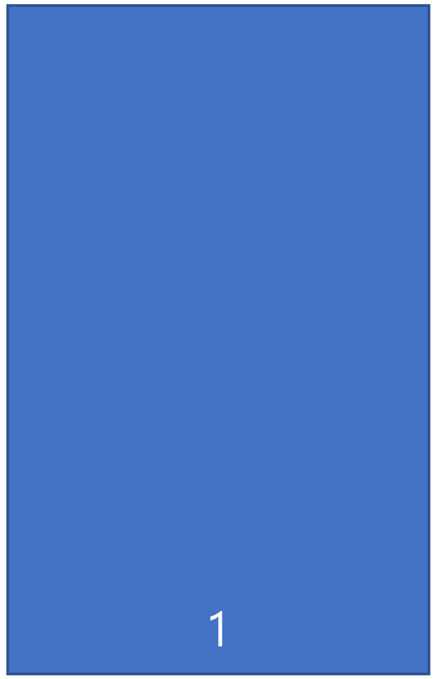
단점

- 단순 검색 속도는 너비 우선 탐색(BFS)보다 느림
- 얻어진 해가 최단 경로가 된다는 보장이 없음

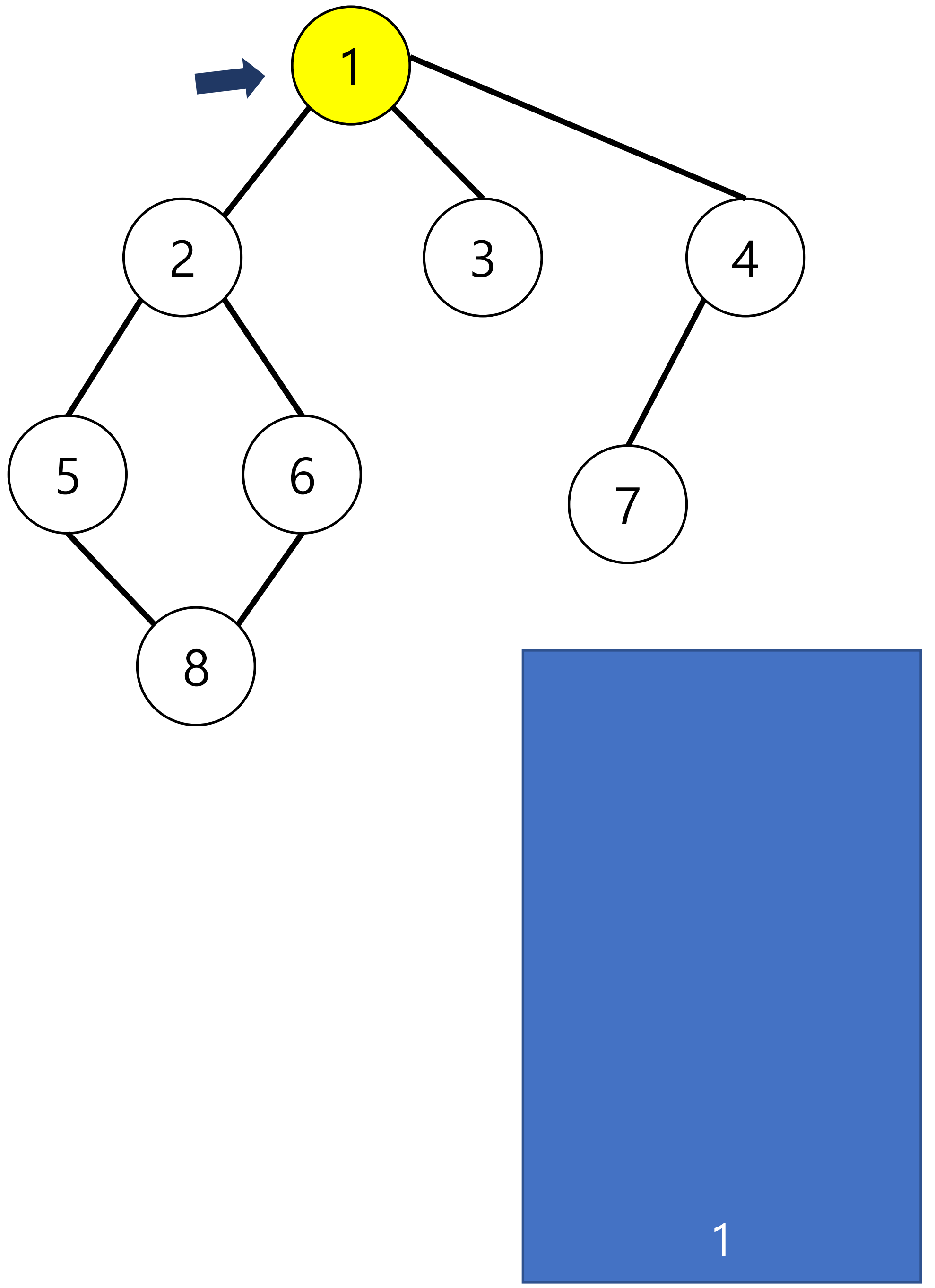


```python
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]
visited = [False] * len(graph)
stack = []

stack.append(1)
visited[1] = True

while len(stack) > 0:
    current_node = None
    for node in graph[stack[-1]]:
        if visited[node] == False:
            current_node = node
            break
    if current_node == None:
        stack.pop()
    else:
        visited[current_node] = True
        print(current_node)
        stack.append(current_node)
```

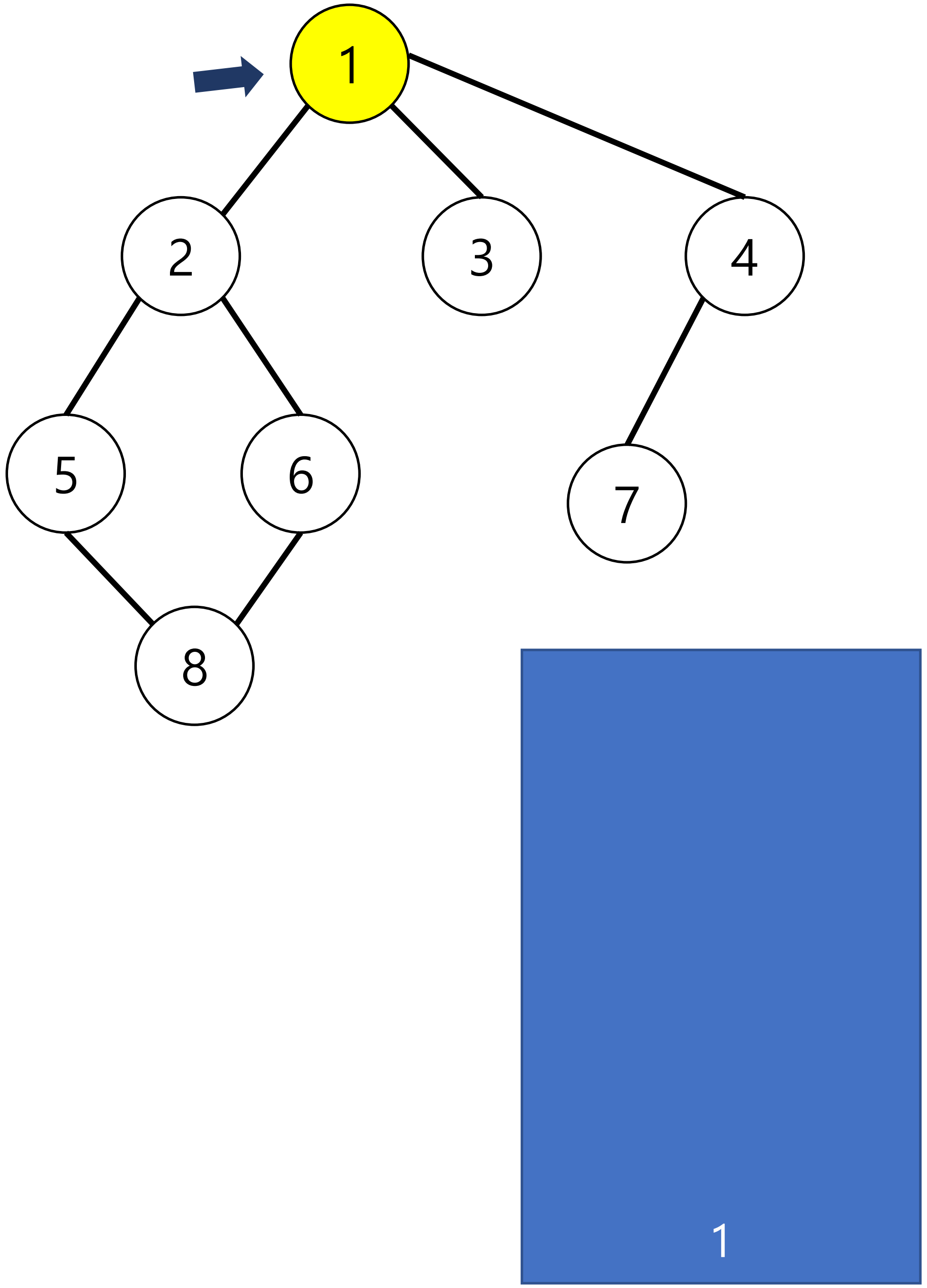




```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]
visited = [False] * len(graph)
stack = []

stack.append(1)
visited[1] = True

while len(stack) > 0:
    current_node = None
    for node in graph[stack[-1]]:
        if visited[node] == False:
            current_node = node
            break
    if current_node == None:
        stack.pop()
    else:
        visited[current_node] = True
        print(current_node)
        stack.append(current_node)
```

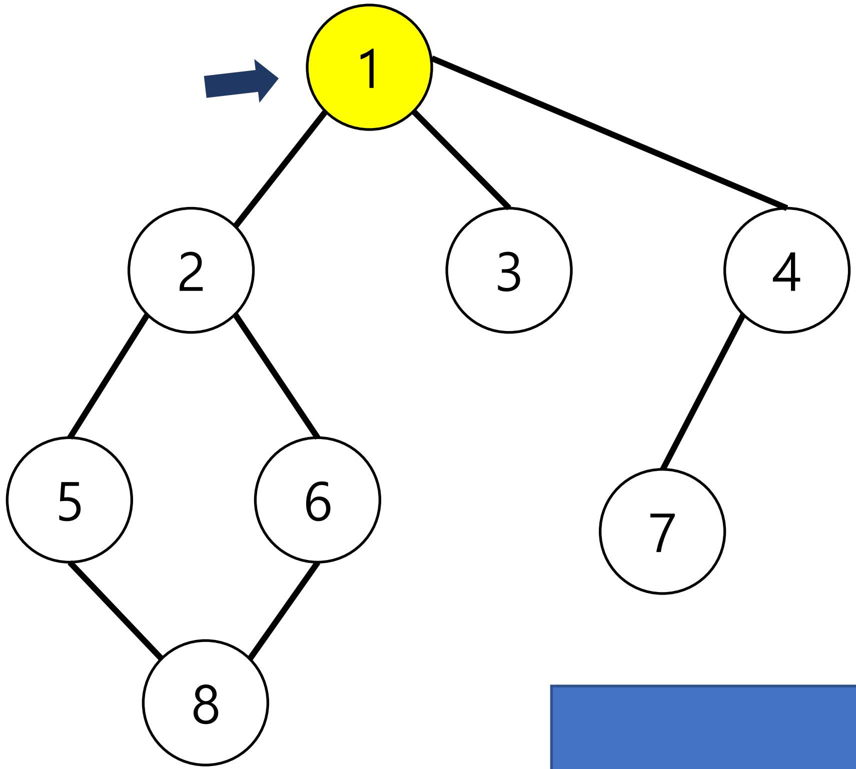


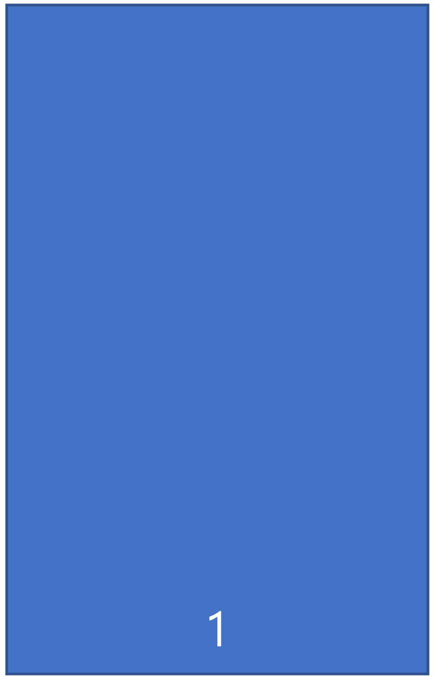
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]
visited = [False] * len(graph)
stack = []

stack.append(1)
visited[1] = True

while len(stack) > 0:
    current_node = None
    for node in graph[stack[-1]]:
        if visited[node] == False:
            current_node = node
            break
    if current_node == None:
        stack.pop()
    else:
        visited[current_node] = True
        print(current_node)
        stack.append(current_node)
```

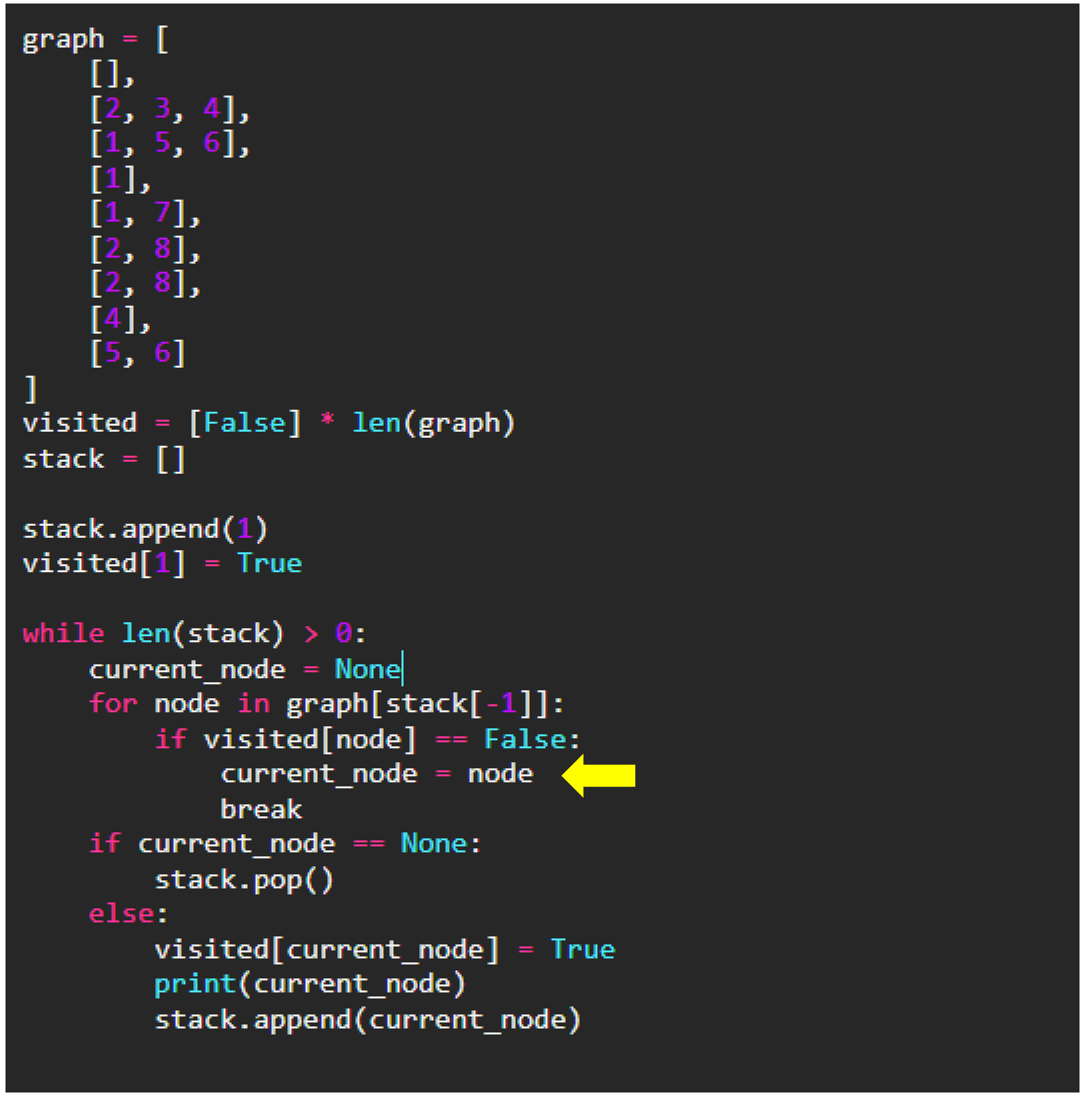


```python
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]
visited = [False] * len(graph)
stack = []

stack.append(1)
visited[1] = True

while len(stack) > 0:
    current_node = None
    for node in graph[stack[-1]]:
        if visited[node] == False:
            current_node = node
            break
    if current_node == None:
        stack.pop()
    else:
        visited[current_node] = True
        print(current_node)
        stack.append(current_node)
```

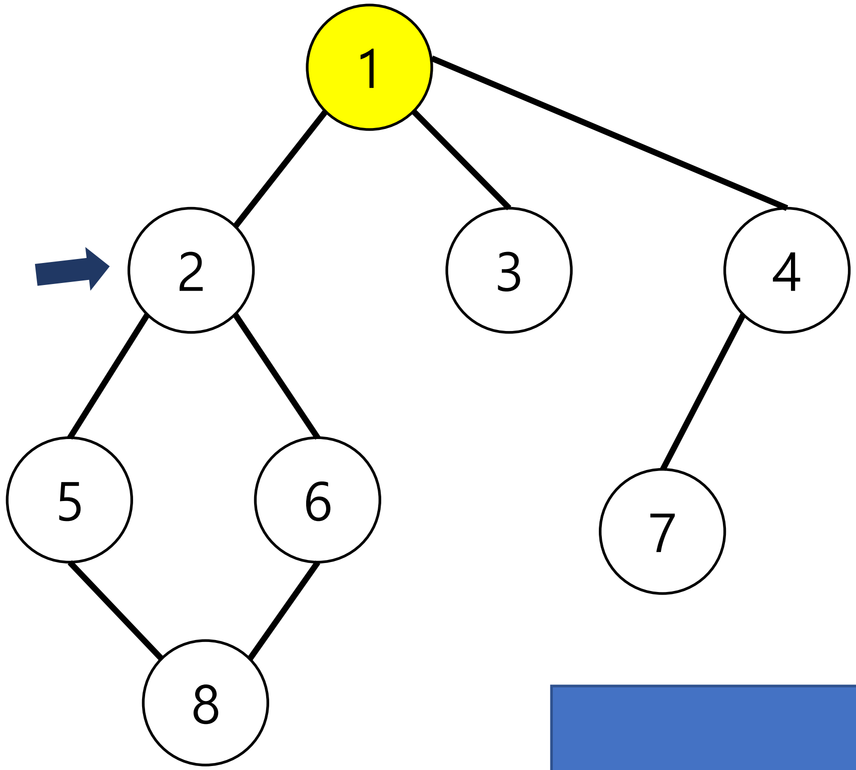
```javascript
for(i=0;i<graph[stack[-1]].length;i++){
    node = graph[stack[-1]][i];
}
```

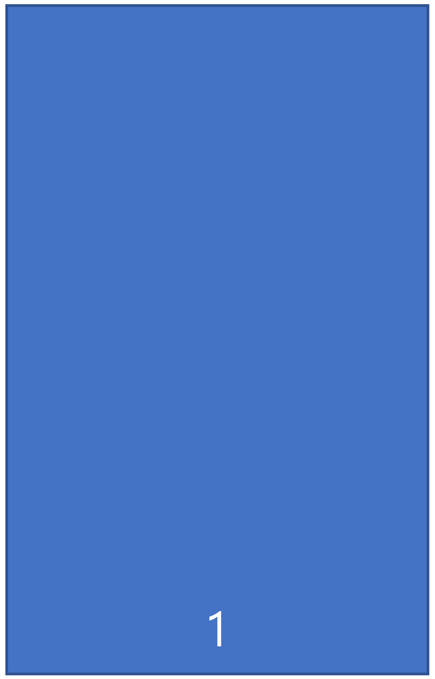


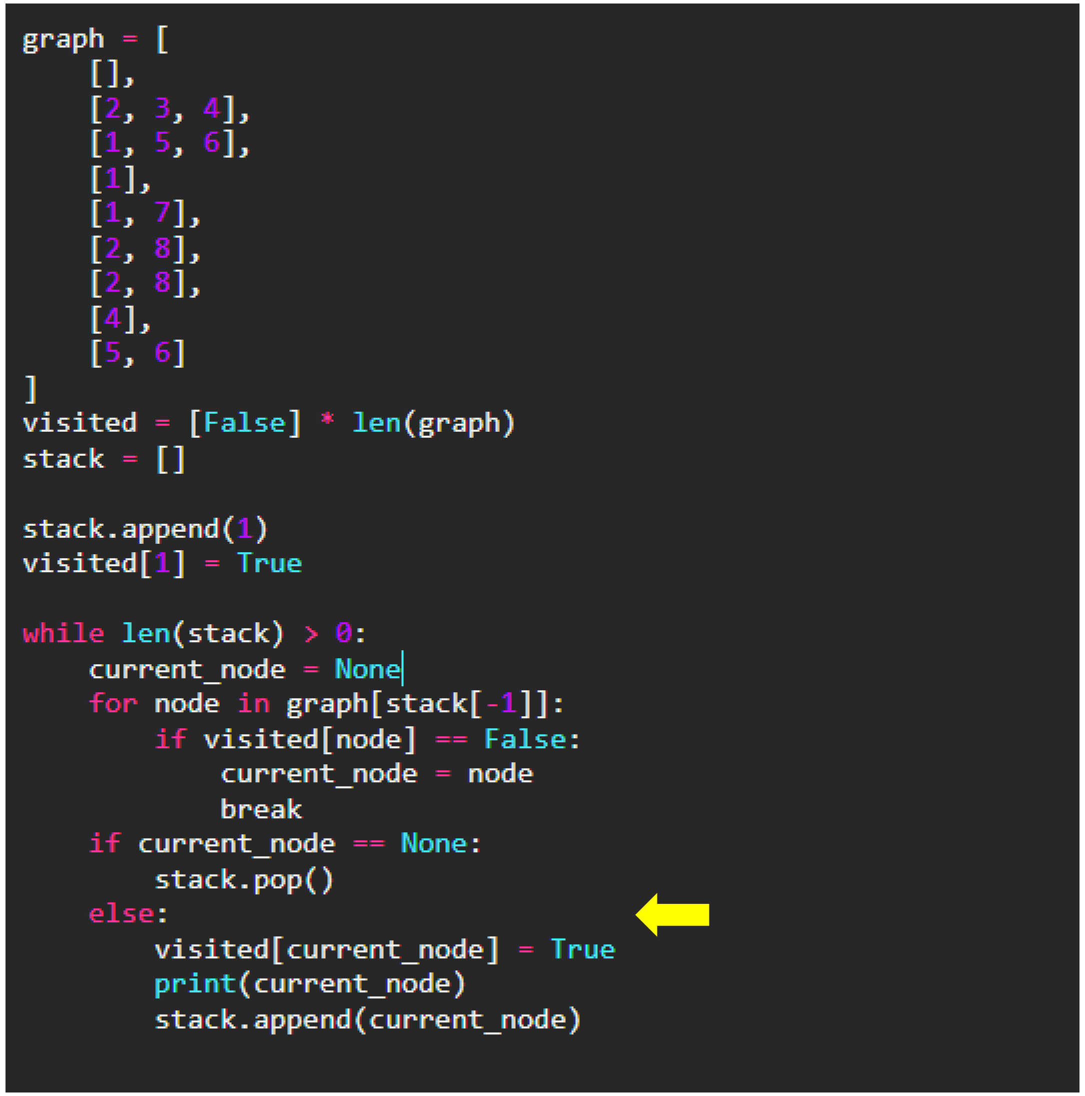




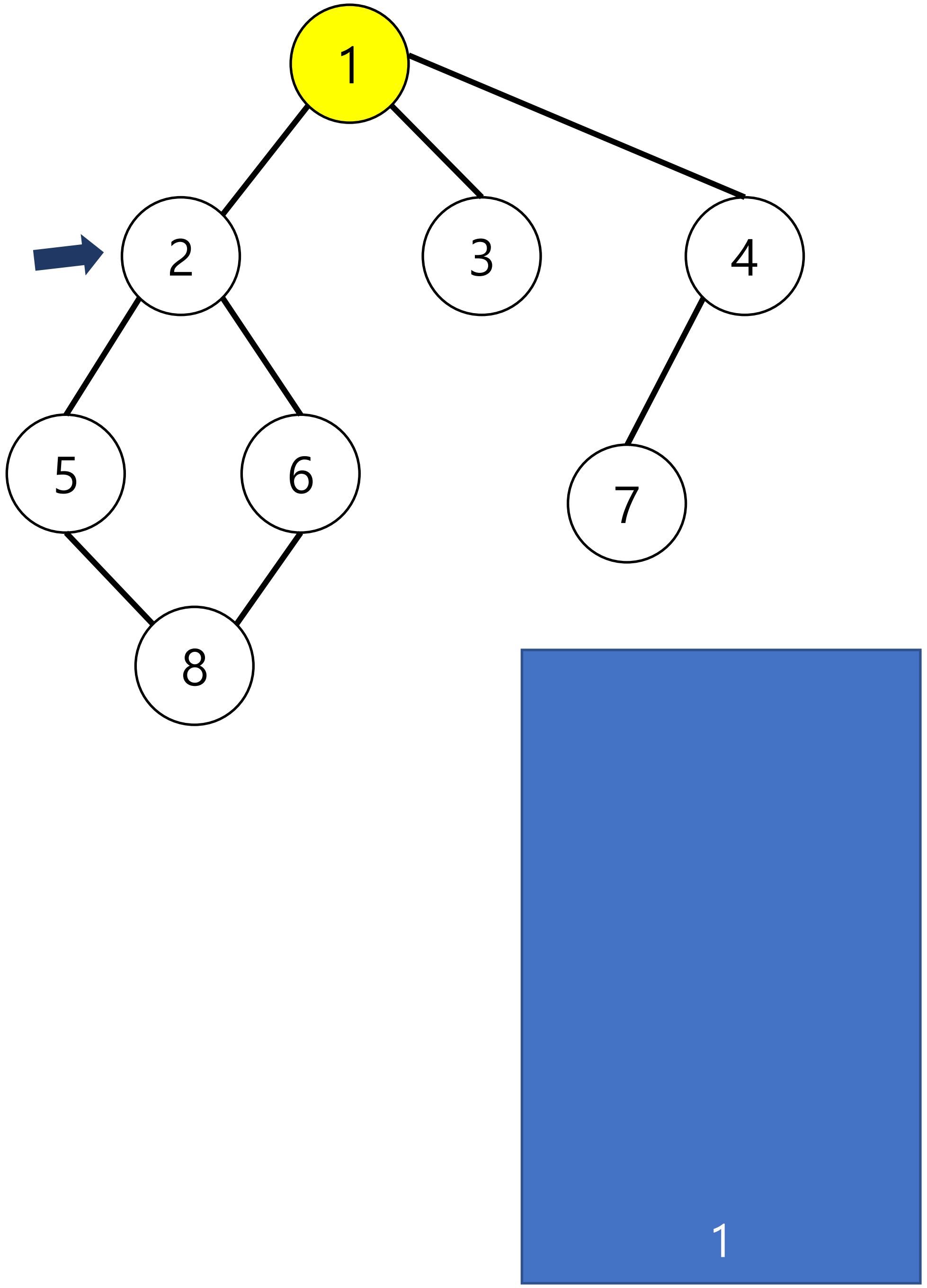
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]
visited = [False] * len(graph)
stack = []

stack.append(1)
visited[1] = True

while len(stack) > 0:
    current_node = None
    for node in graph[stack[-1]]:
        if visited[node] == False:
            current_node = node
            break
    if current_node == None:
        stack.pop()
    else:
        visited[current_node] = True
        print(current_node)
        stack.append(current_node)
```

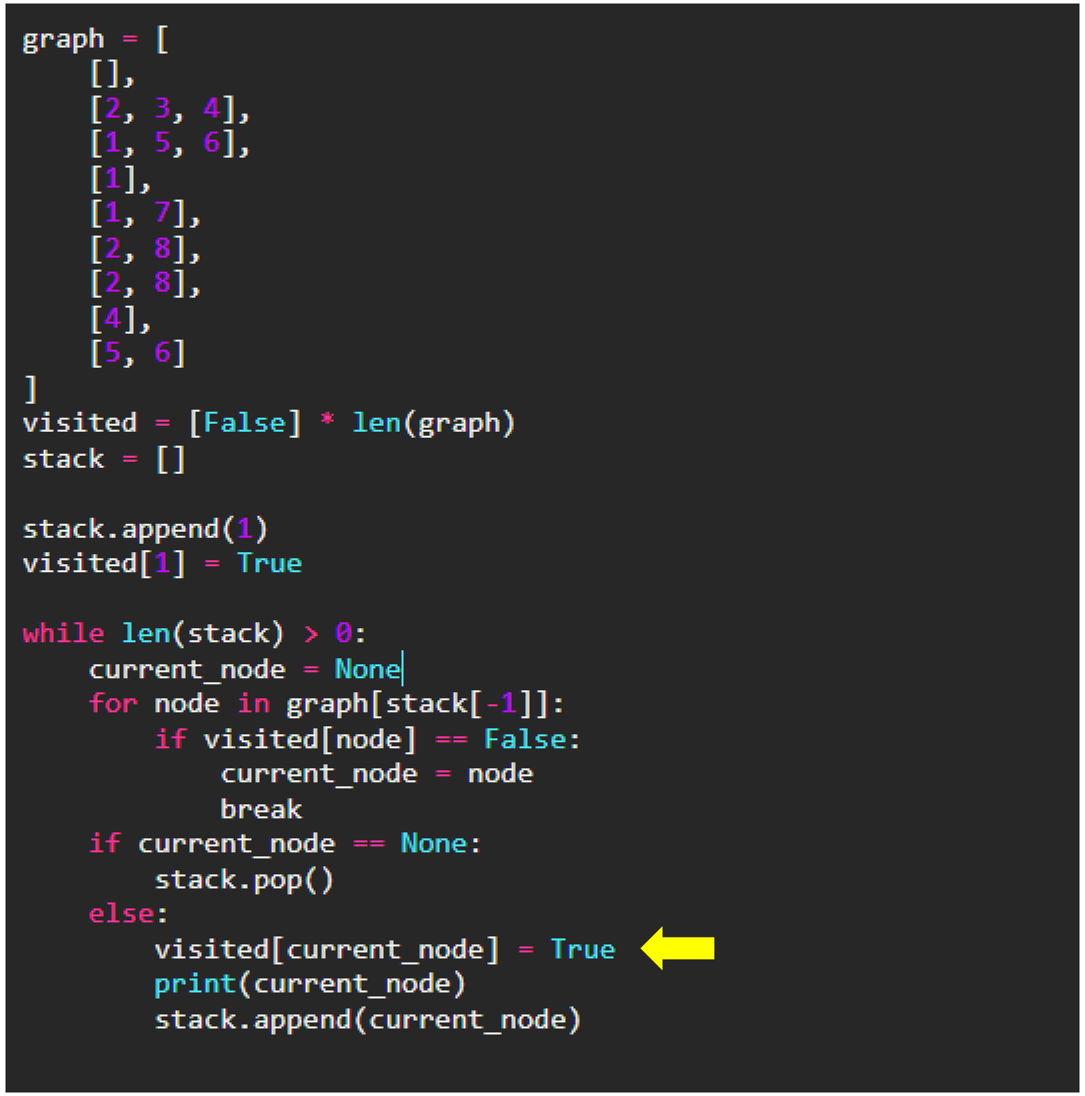




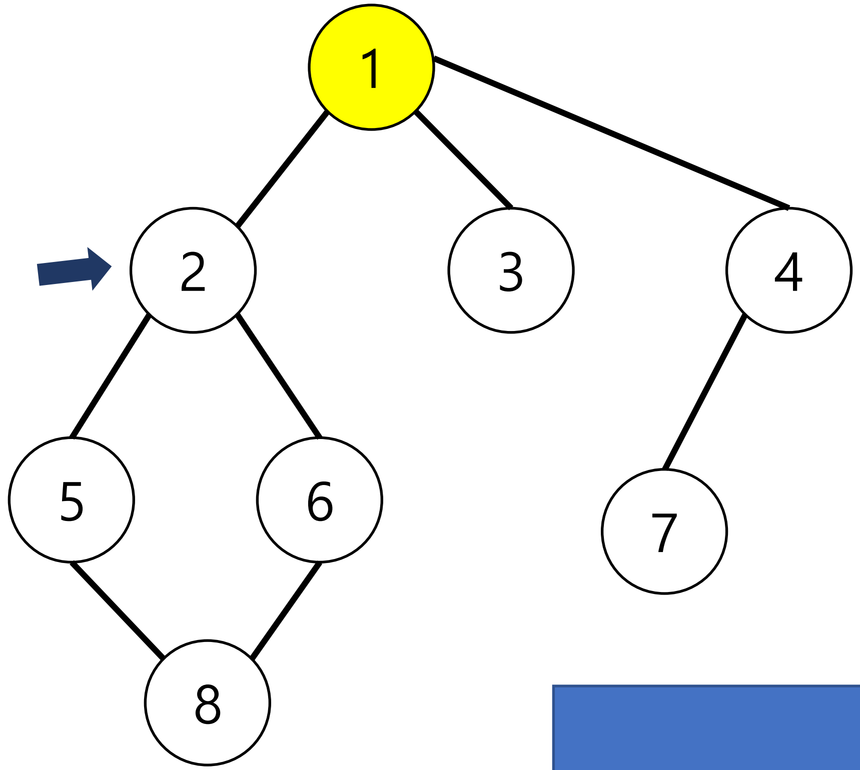


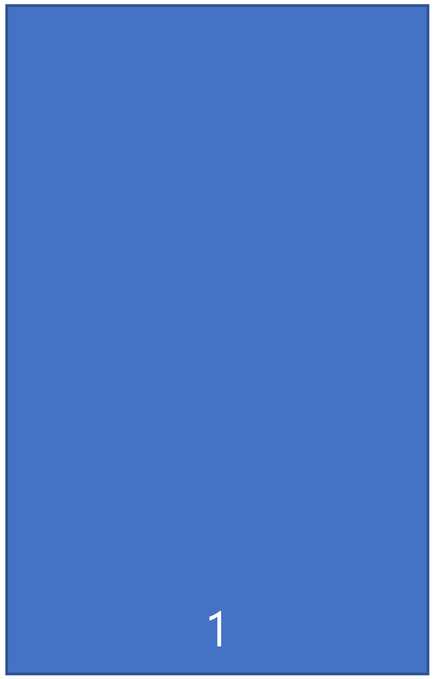
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]
visited = [False] * len(graph)
stack = []

stack.append(1)
visited[1] = True

while len(stack) > 0:
    current_node = None
    for node in graph[stack[-1]]:
        if visited[node] == False:
            current_node = node
            break
    if current_node == None:
        stack.pop()
    else:
        visited[current_node] = True
        print(current_node)
        stack.append(current_node)
```



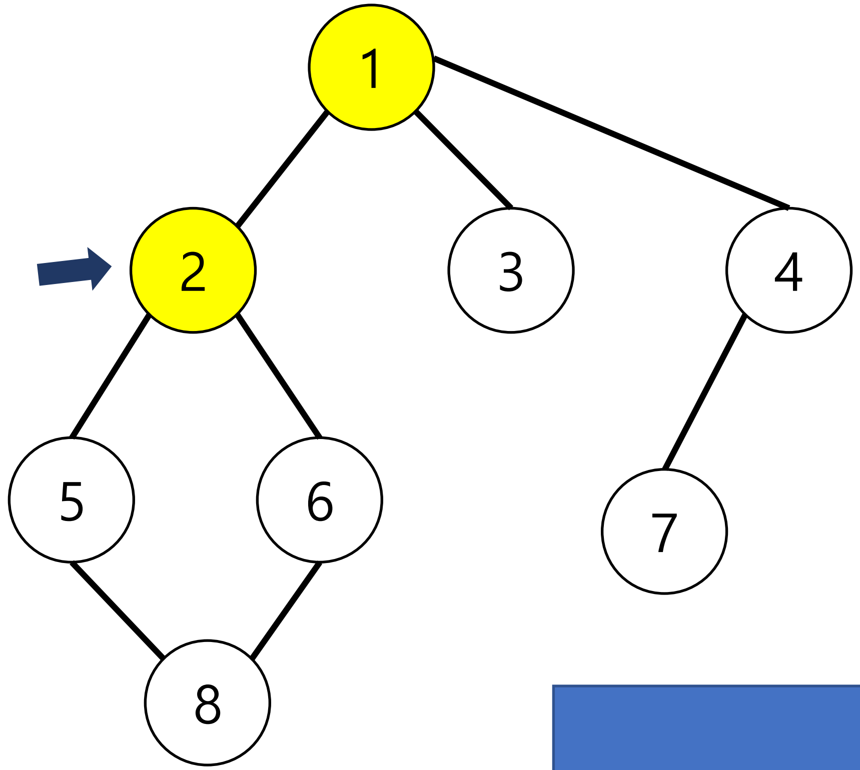


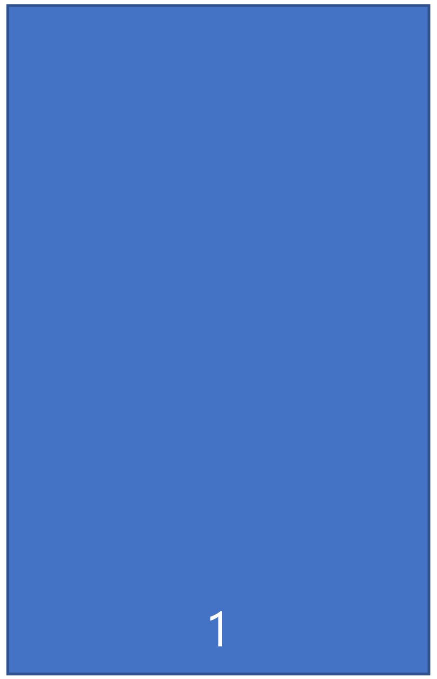
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]
visited = [False] * len(graph)
stack = []

stack.append(1)
visited[1] = True

while len(stack) > 0:
    current_node = None
    for node in graph[stack[-1]]:
        if visited[node] == False:
            current_node = node
            break
    if current_node == None:
        stack.pop()
    else:
        visited[current_node] = True
        print(current_node)
        stack.append(current_node)
```



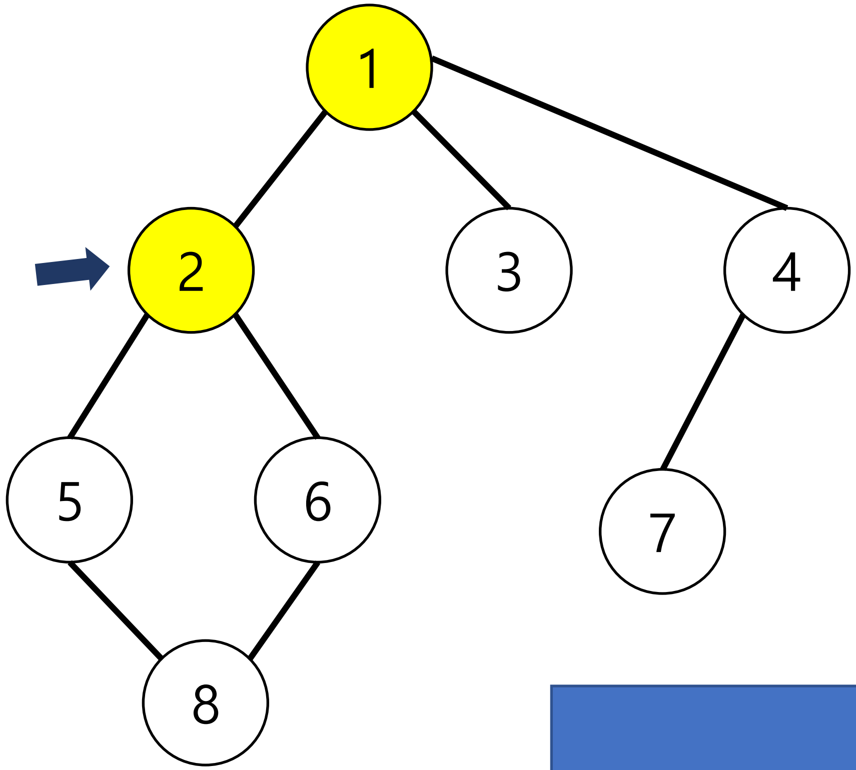


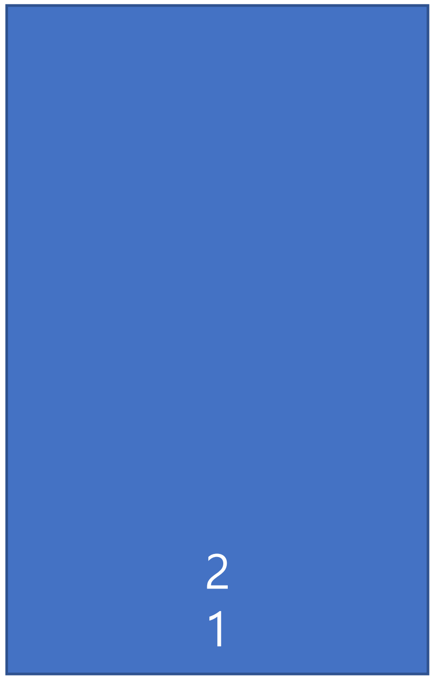
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]
visited = [False] * len(graph)
stack = []

stack.append(1)
visited[1] = True

while len(stack) > 0:
    current_node = None
    for node in graph[stack[-1]]:
        if visited[node] == False:
            current_node = node
            break
    if current_node == None:
        stack.pop()
    else:
        visited[current_node] = True
        print(current_node)
        stack.append(current_node)
```



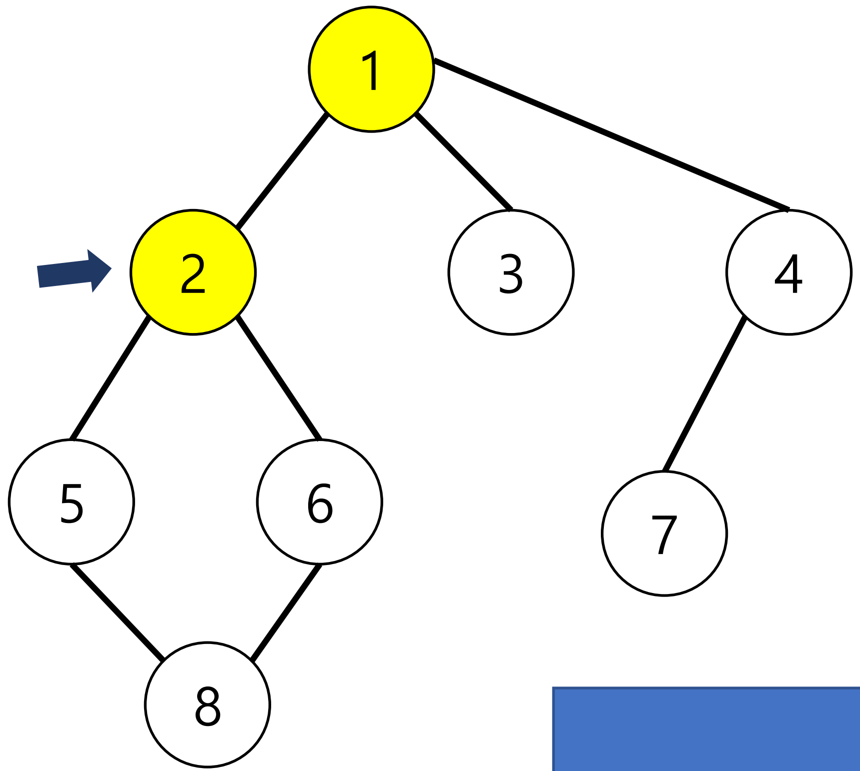


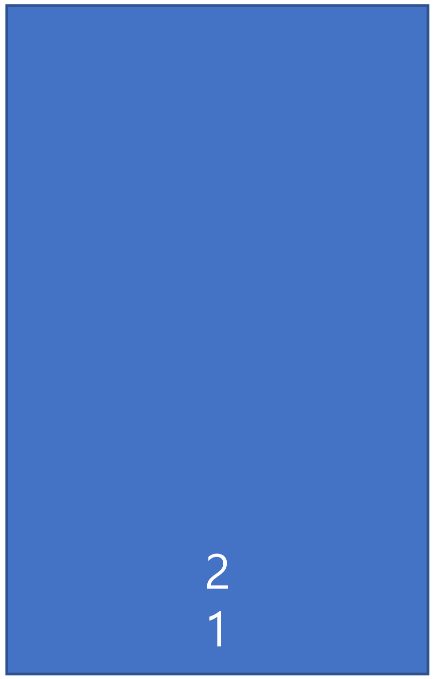
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]
visited = [False] * len(graph)
stack = []

stack.append(1)
visited[1] = True

while len(stack) > 0:
    current_node = None
    for node in graph[stack[-1]]:
        if visited[node] == False:
            current_node = node
            break
    if current_node == None:
        stack.pop()
    else:
        visited[current_node] = True
        print(current_node)
        stack.append(current_node)
```

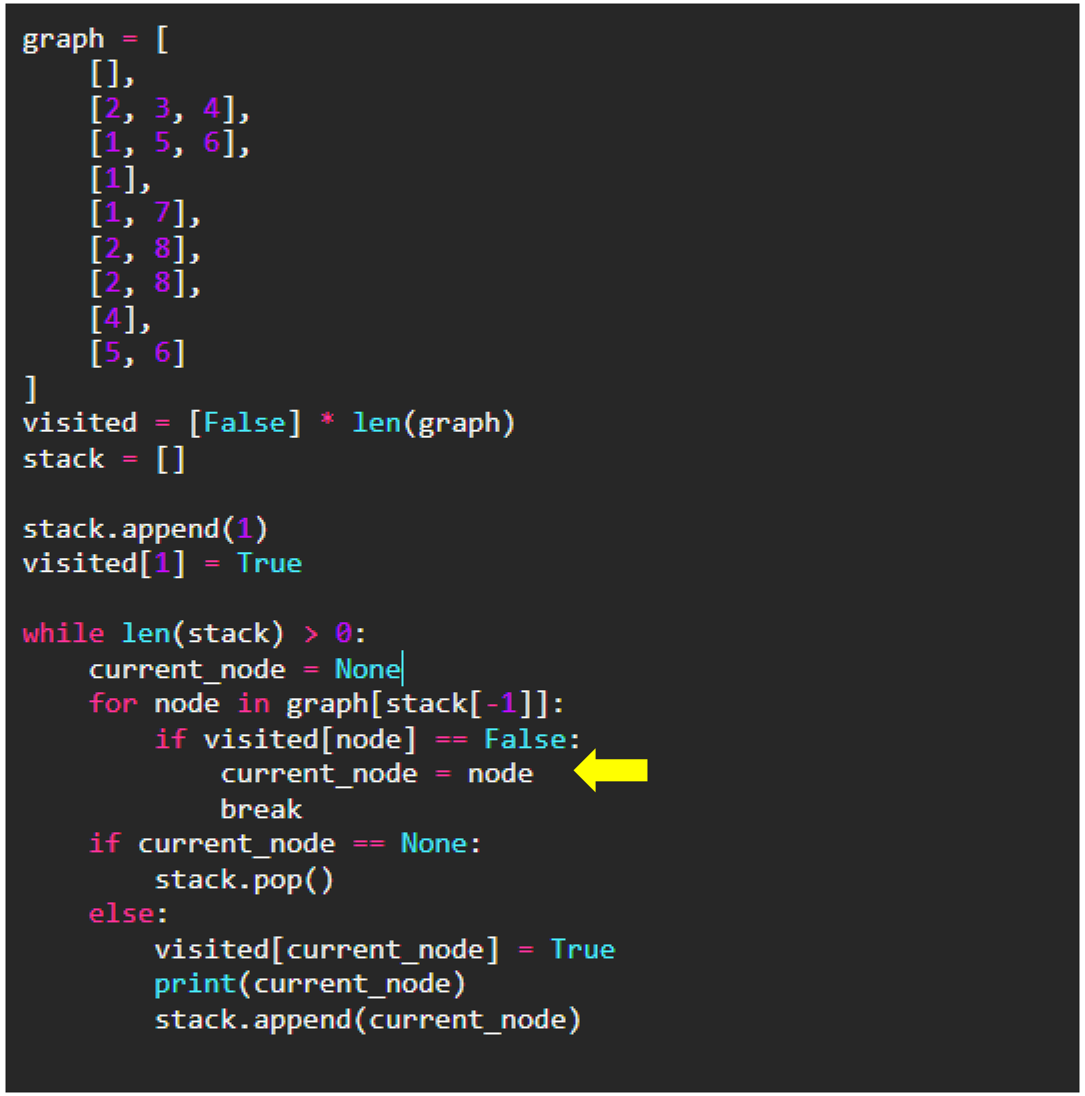




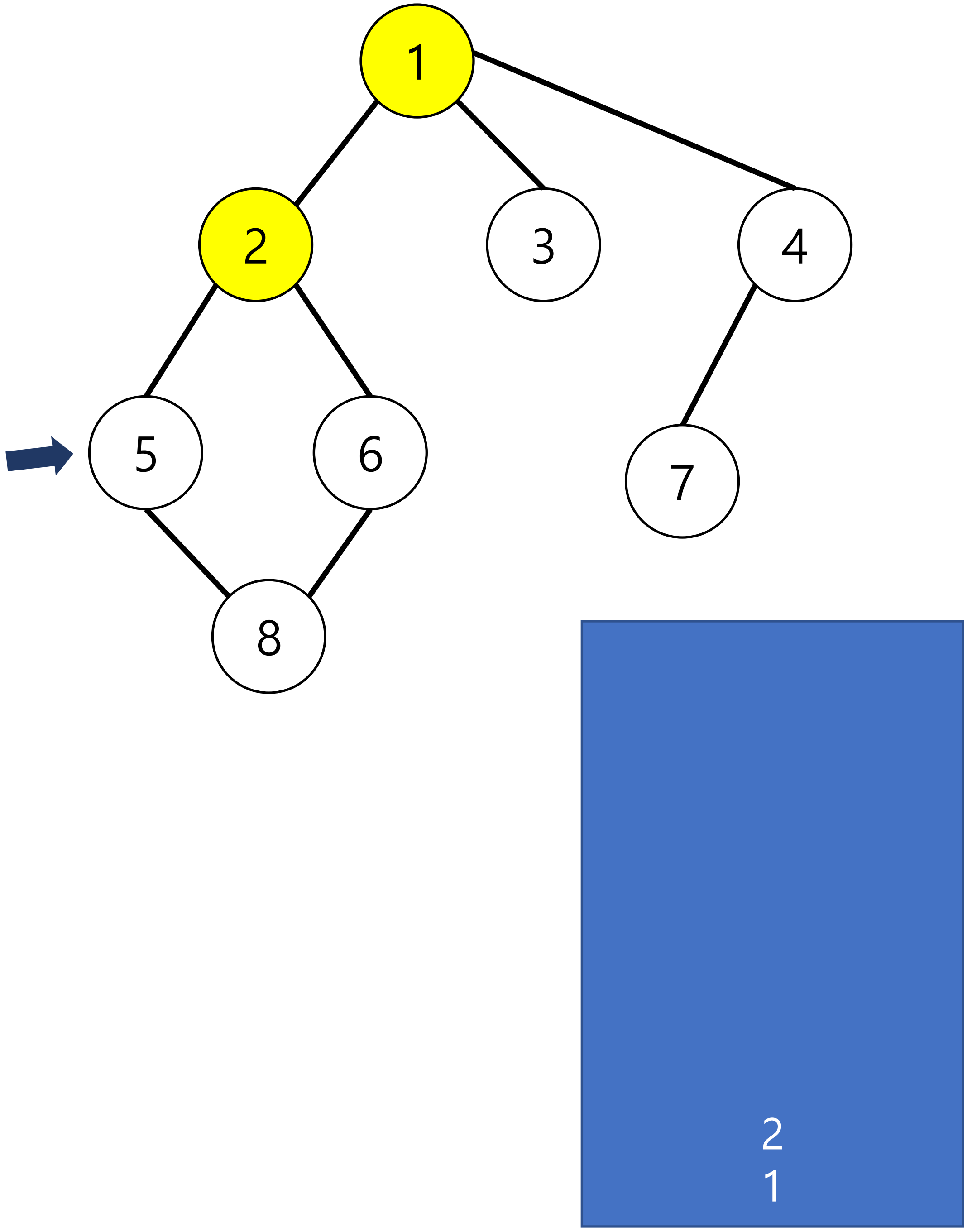
```python
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]
visited = [False] * len(graph)
stack = []

stack.append(1)
visited[1] = True

while len(stack) > 0:
    current_node = None
    for node in graph[stack[-1]]:
        if visited[node] == False:
            current_node = node
            break
    if current_node == None:
        stack.pop()
    else:
        visited[current_node] = True
        print(current_node)
        stack.append(current_node)
```

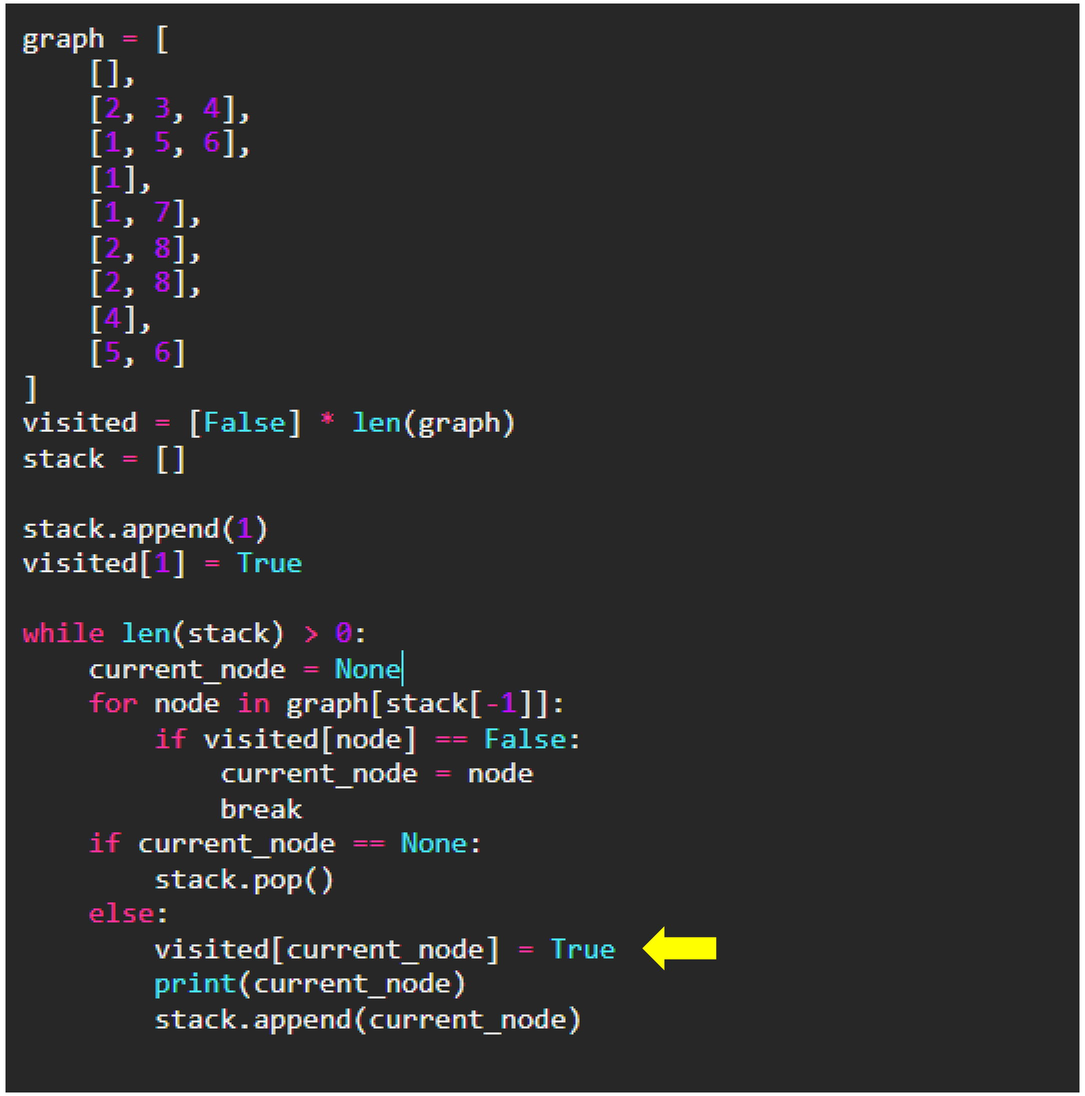




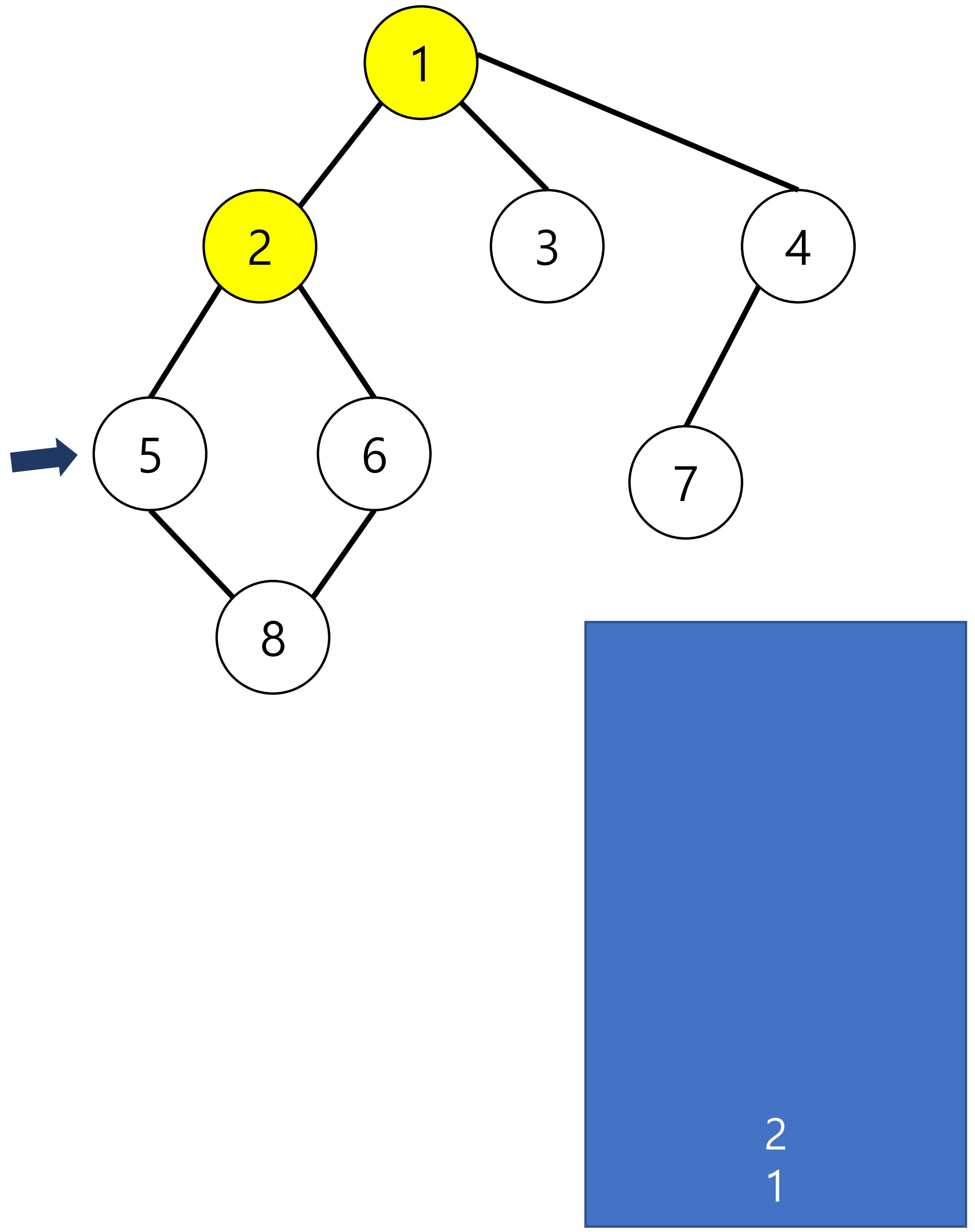


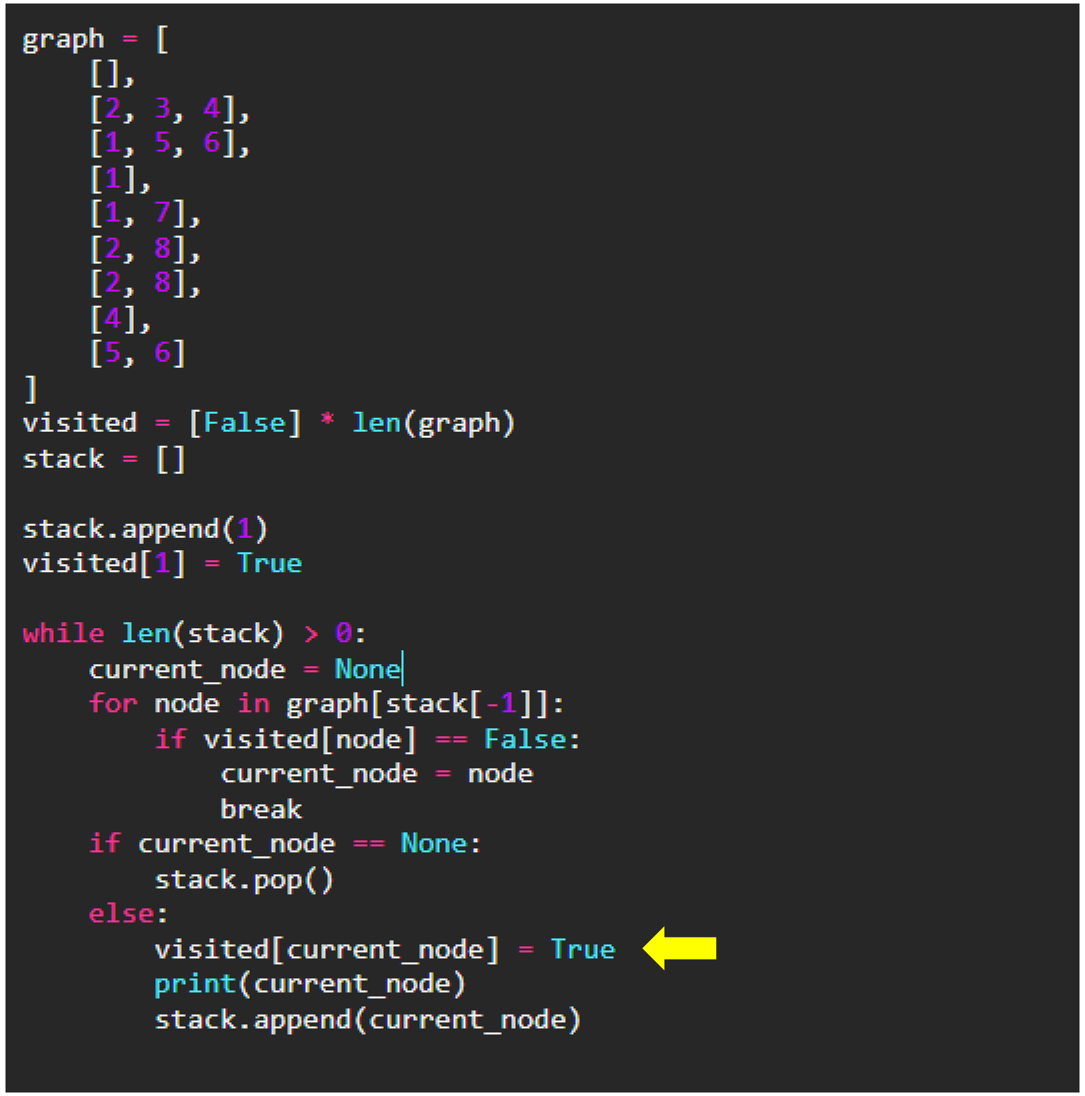
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]
visited = [False] * len(graph)
stack = []

stack.append(1)
visited[1] = True

while len(stack) > 0:
    current_node = None
    for node in graph[stack[-1]]:
        if visited[node] == False:
            current_node = node
            break
    if current_node == None:
        stack.pop()
    else:
        visited[current_node] = True
        print(current_node)
        stack.append(current_node)
```



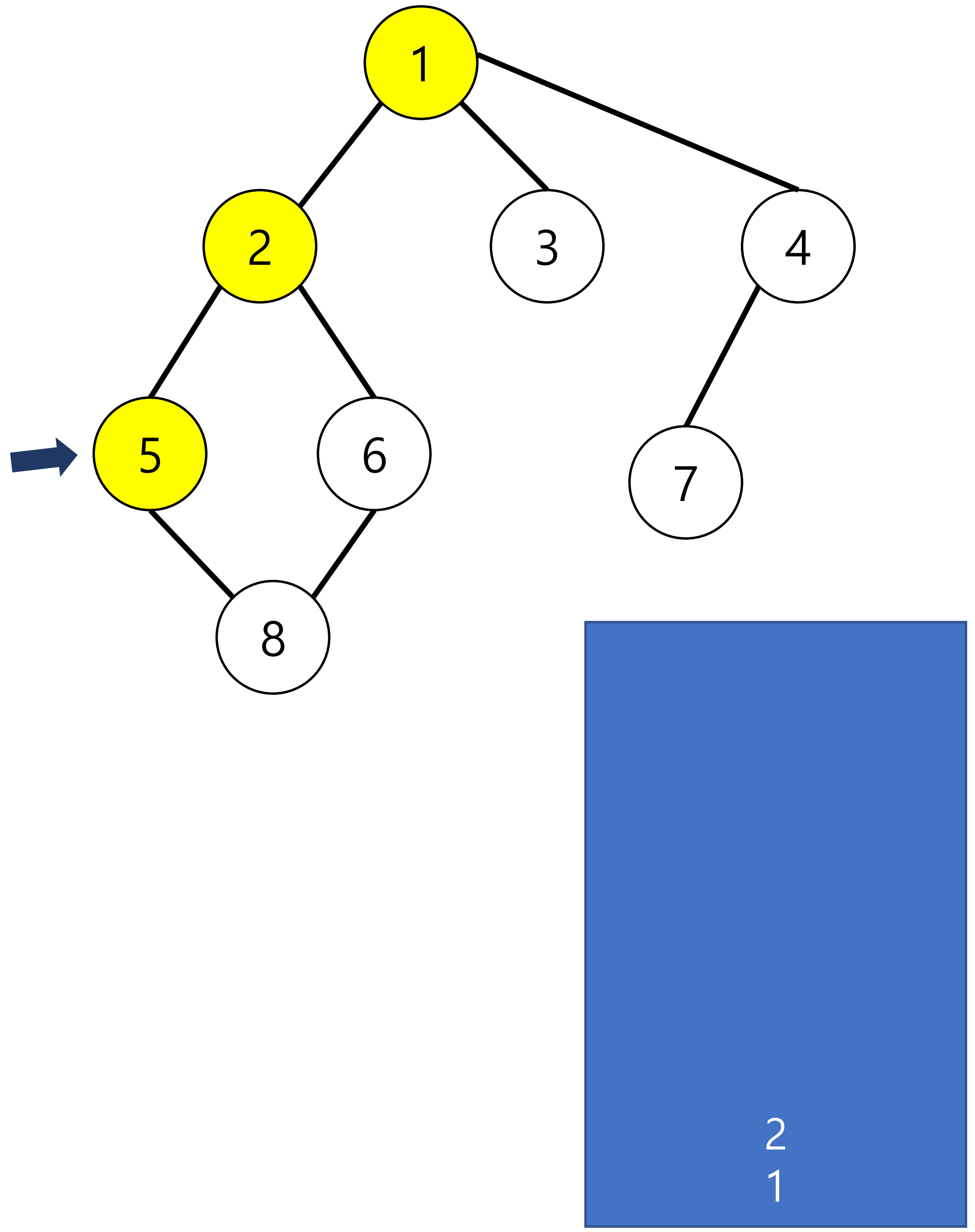


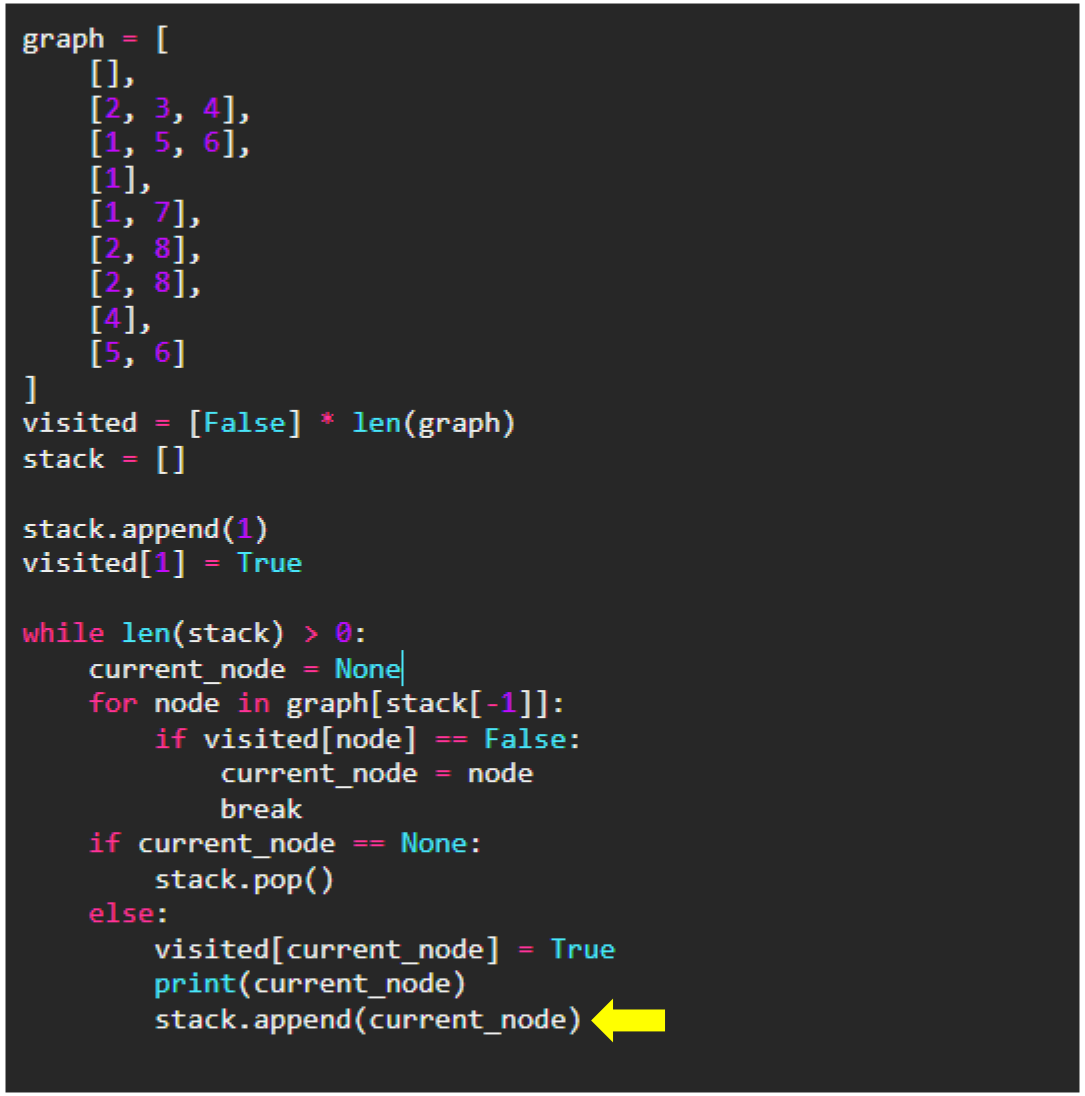
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]
visited = [False] * len(graph)
stack = []

stack.append(1)
visited[1] = True

while len(stack) > 0:
    current_node = None
    for node in graph[stack[-1]]:
        if visited[node] == False:
            current_node = node
            break
    if current_node == None:
        stack.pop()
    else:
        visited[current_node] = True
        print(current_node)
        stack.append(current_node)
```



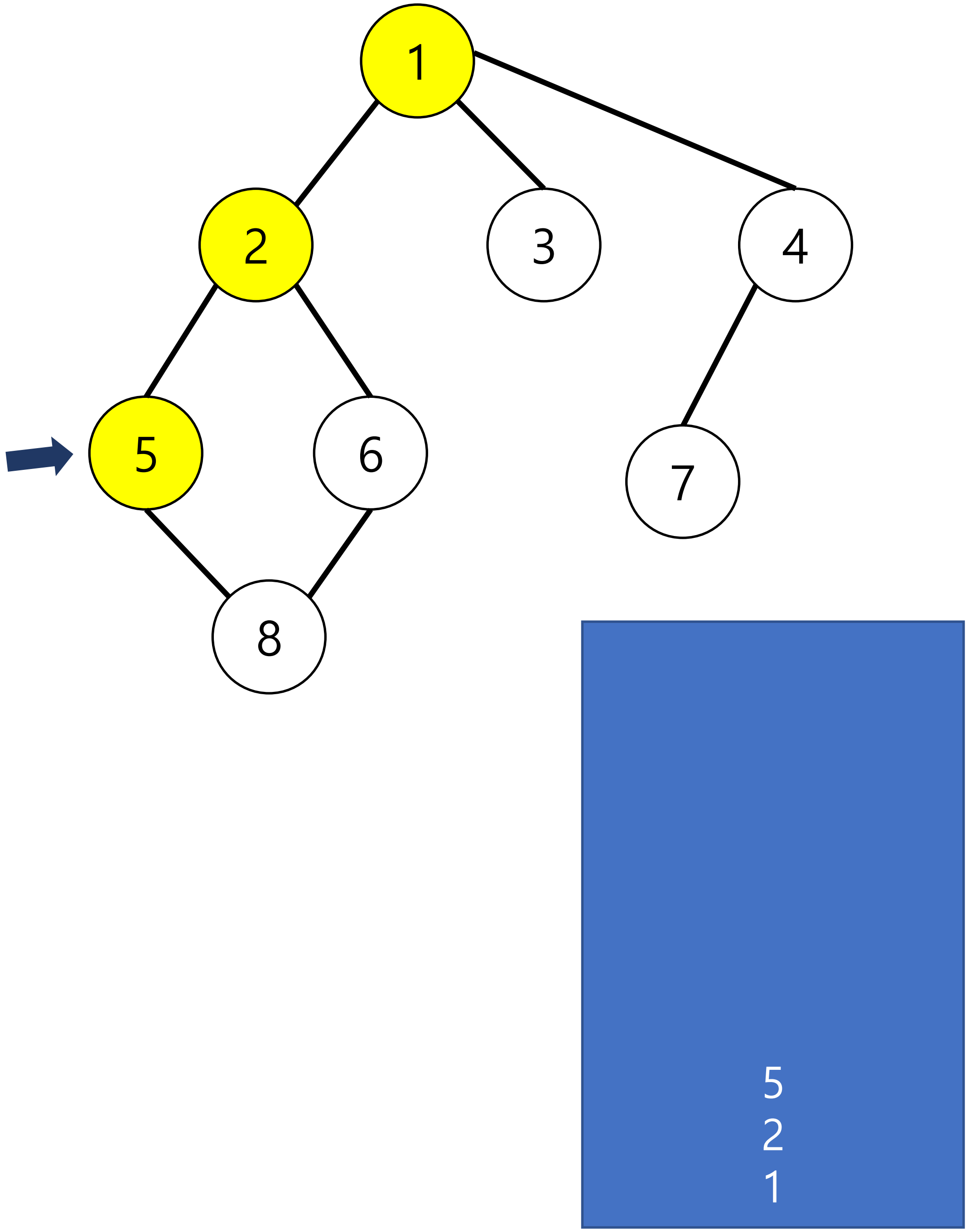


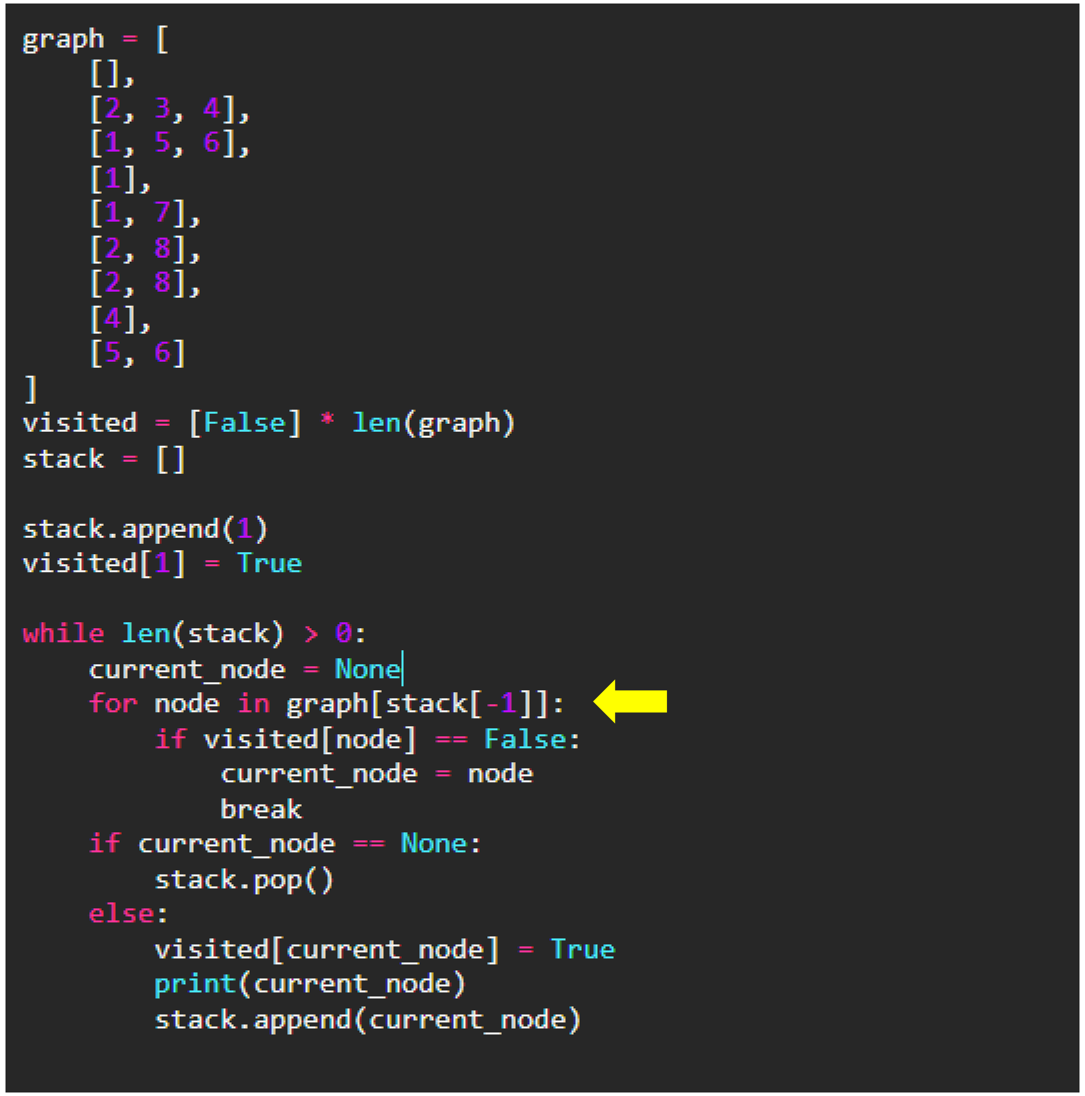
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]
visited = [False] * len(graph)
stack = []

stack.append(1)
visited[1] = True

while len(stack) > 0:
    current_node = None
    for node in graph[stack[-1]]:
        if visited[node] == False:
            current_node = node
            break
    if current_node == None:
        stack.pop()
    else:
        visited[current_node] = True
        print(current_node)
        stack.append(current_node)
```



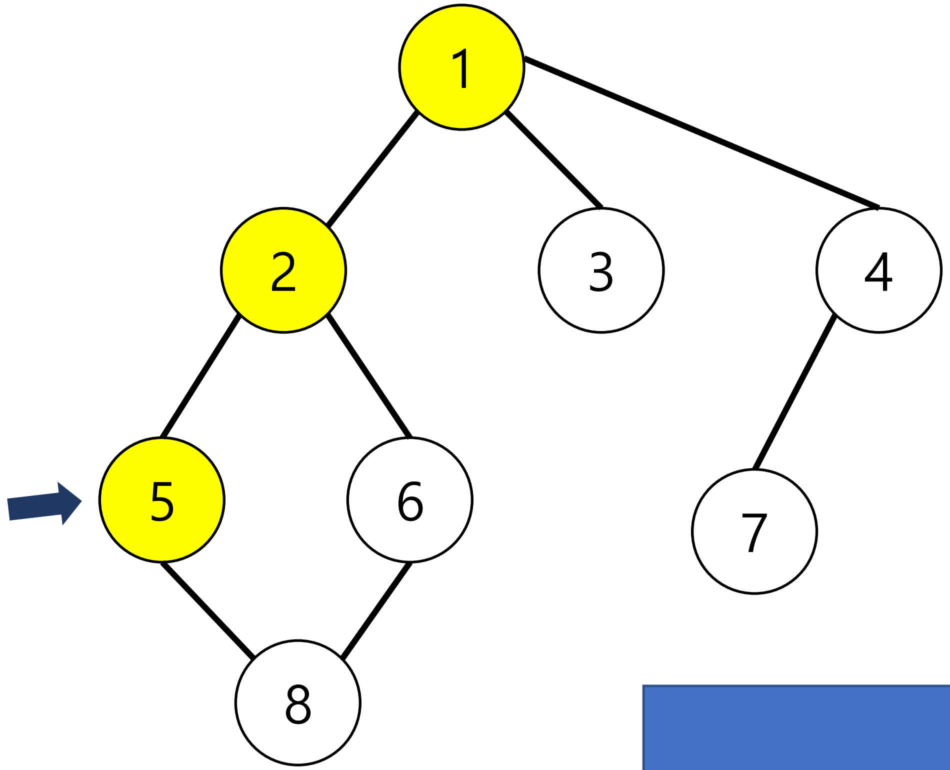


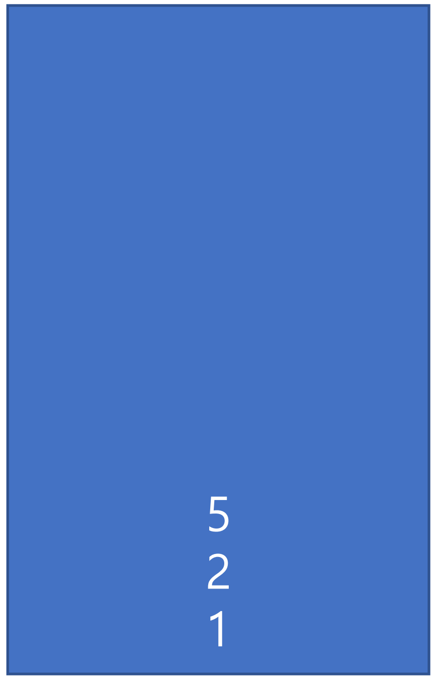
```python
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]
visited = [False] * len(graph)
stack = []

stack.append(1)
visited[1] = True

while len(stack) > 0:
    current_node = None
    for node in graph[stack[-1]]:
        if visited[node] == False:
            current_node = node
            break
    if current_node == None:
        stack.pop()
    else:
        visited[current_node] = True
        print(current_node)
        stack.append(current_node)
```

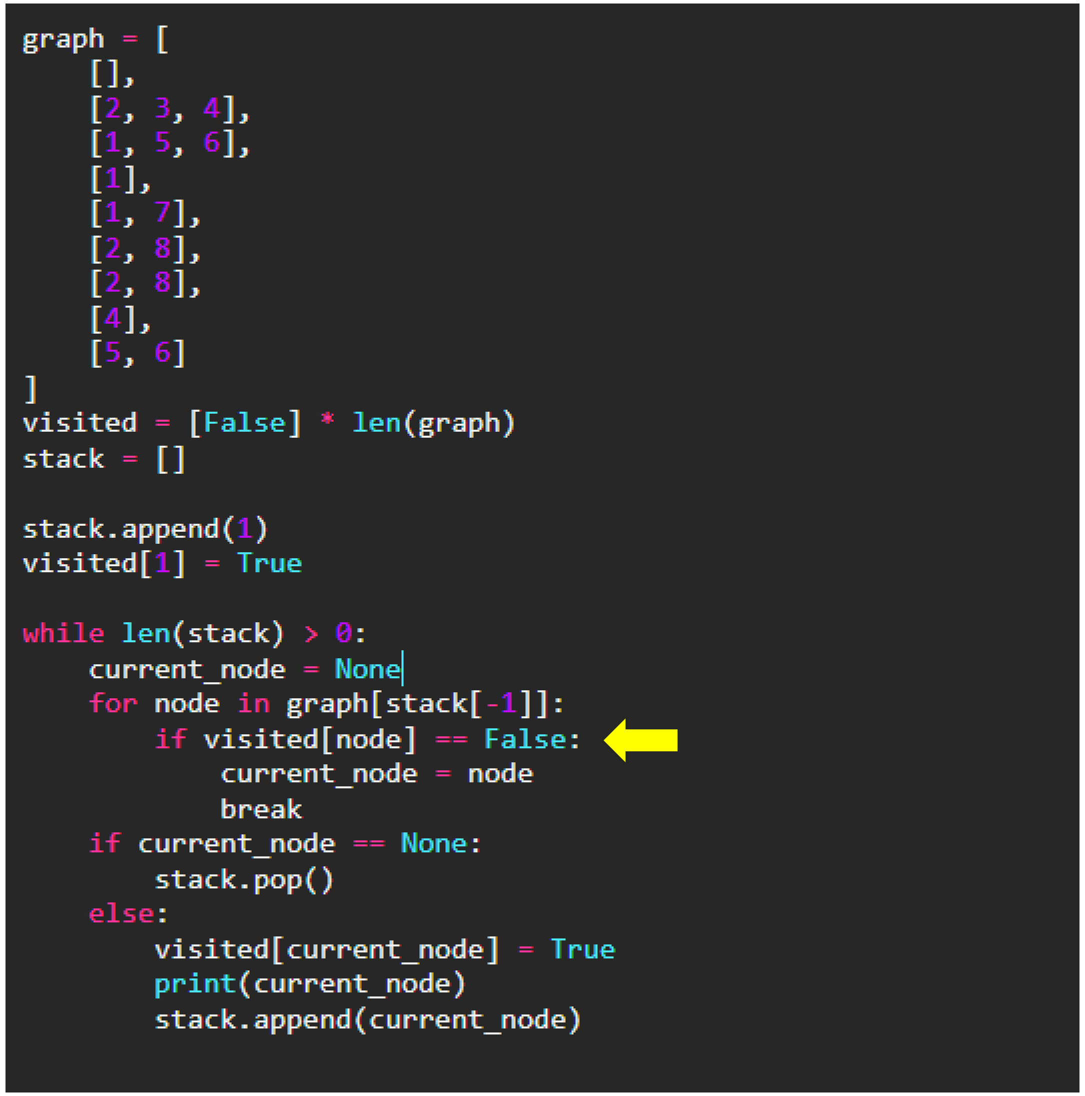




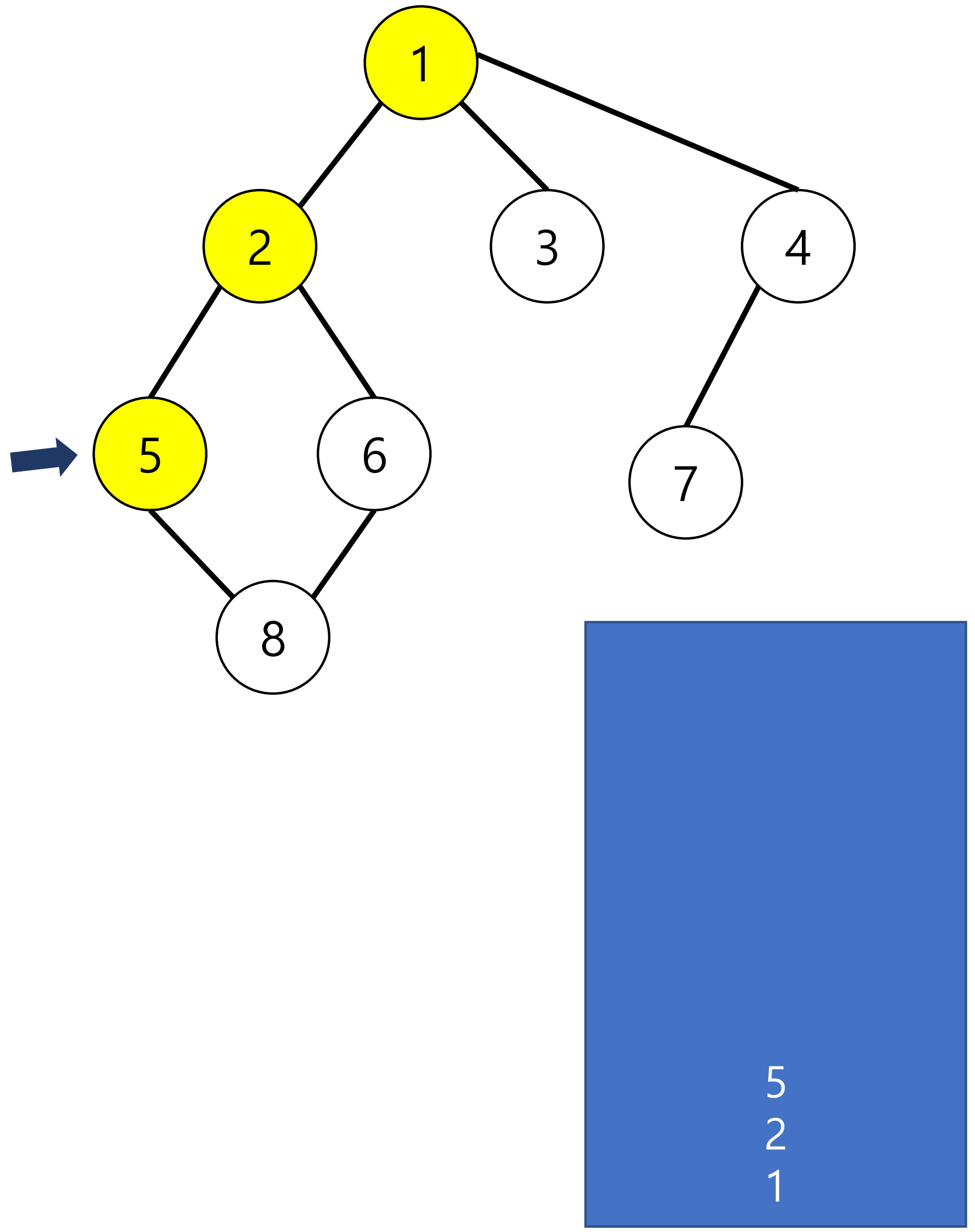
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]
visited = [False] * len(graph)
stack = []

stack.append(1)
visited[1] = True

while len(stack) > 0:
    current_node = None
    for node in graph[stack[-1]]:
        if visited[node] == False:
            current_node = node
            break
    if current_node == None:
        stack.pop()
    else:
        visited[current_node] = True
        print(current_node)
        stack.append(current_node)
```

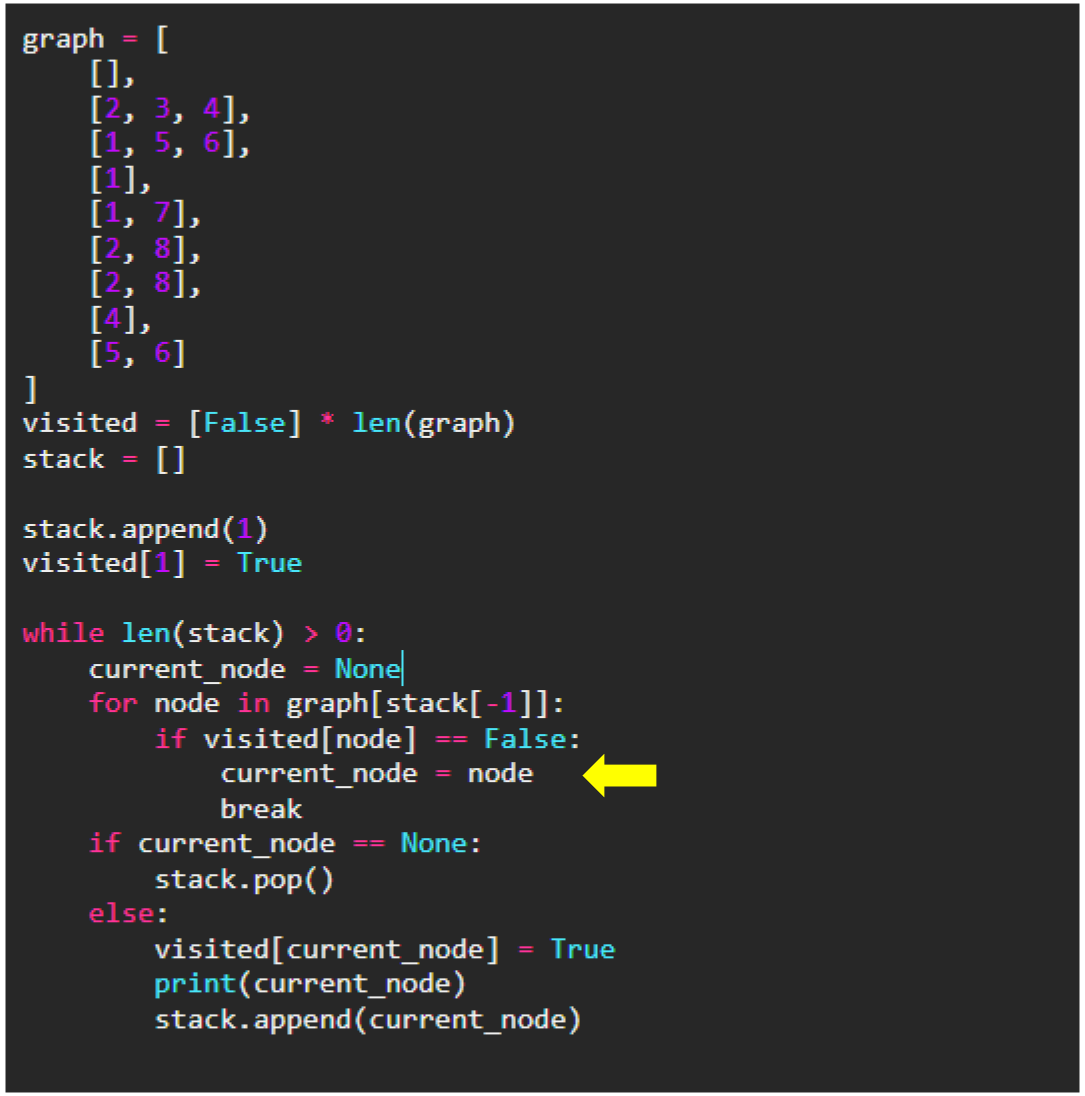




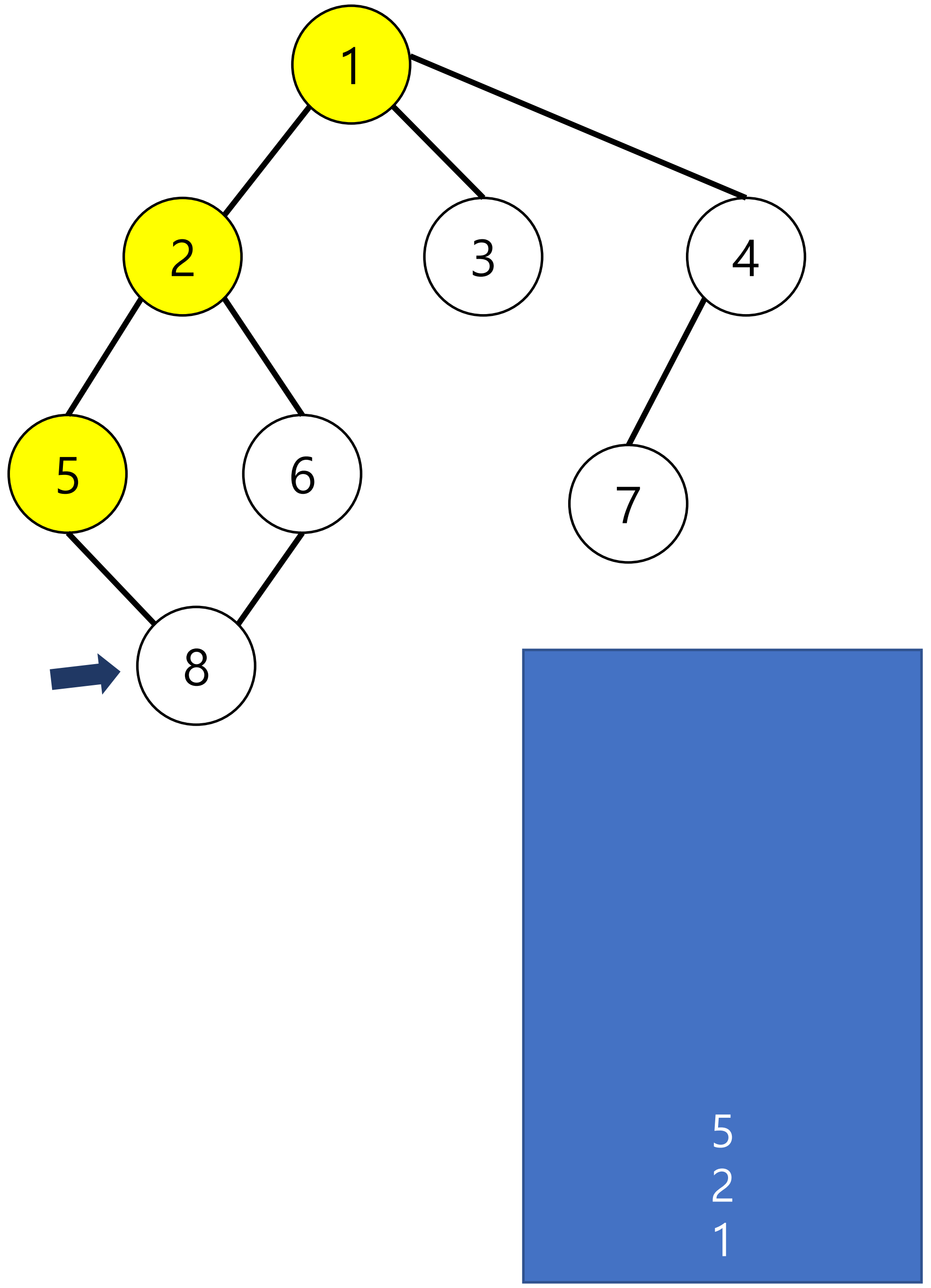


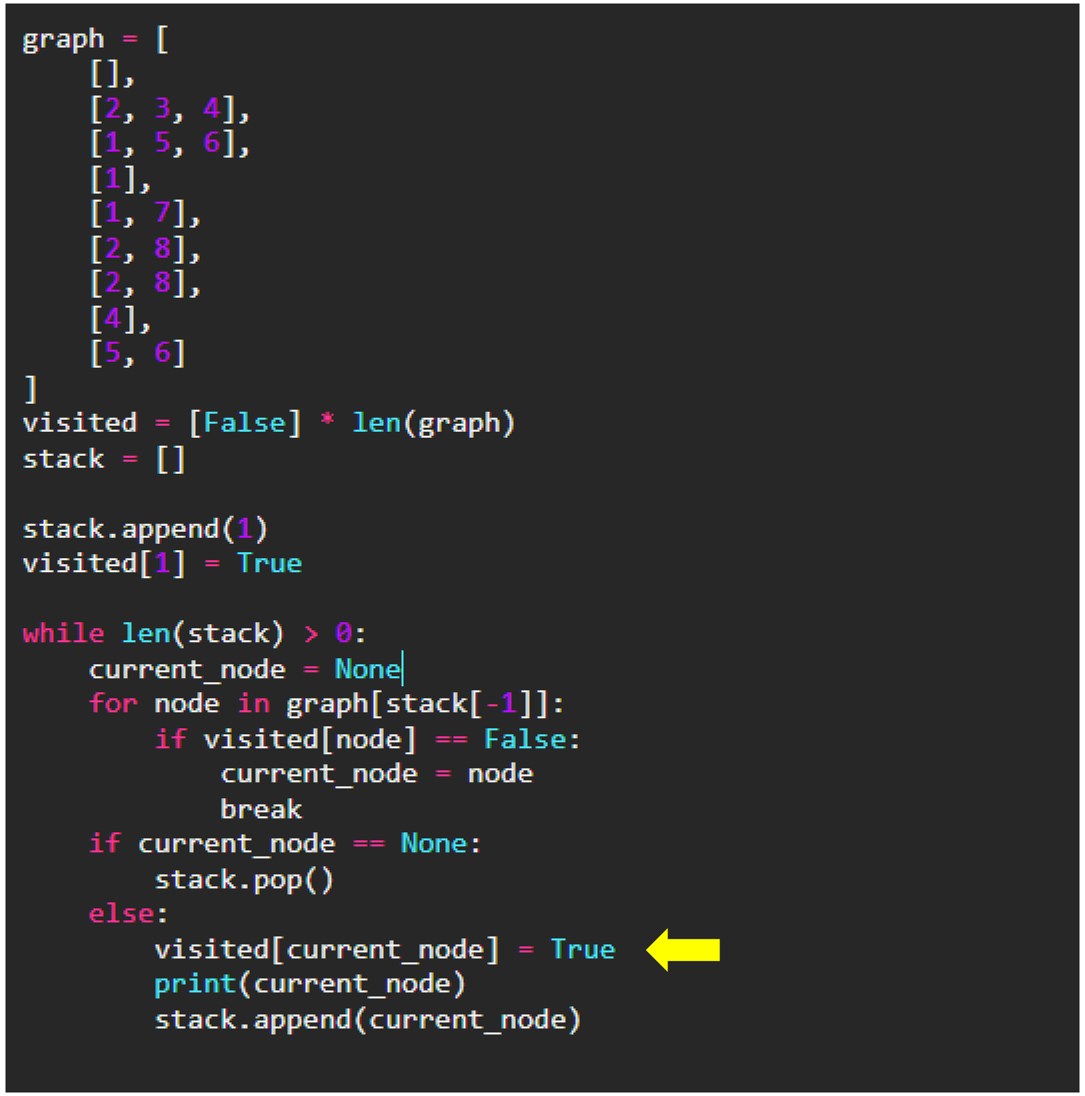
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]
visited = [False] * len(graph)
stack = []

stack.append(1)
visited[1] = True

while len(stack) > 0:
    current_node = None
    for node in graph[stack[-1]]:
        if visited[node] == False:
            current_node = node
            break
    if current_node == None:
        stack.pop()
    else:
        visited[current_node] = True
        print(current_node)
        stack.append(current_node)
```



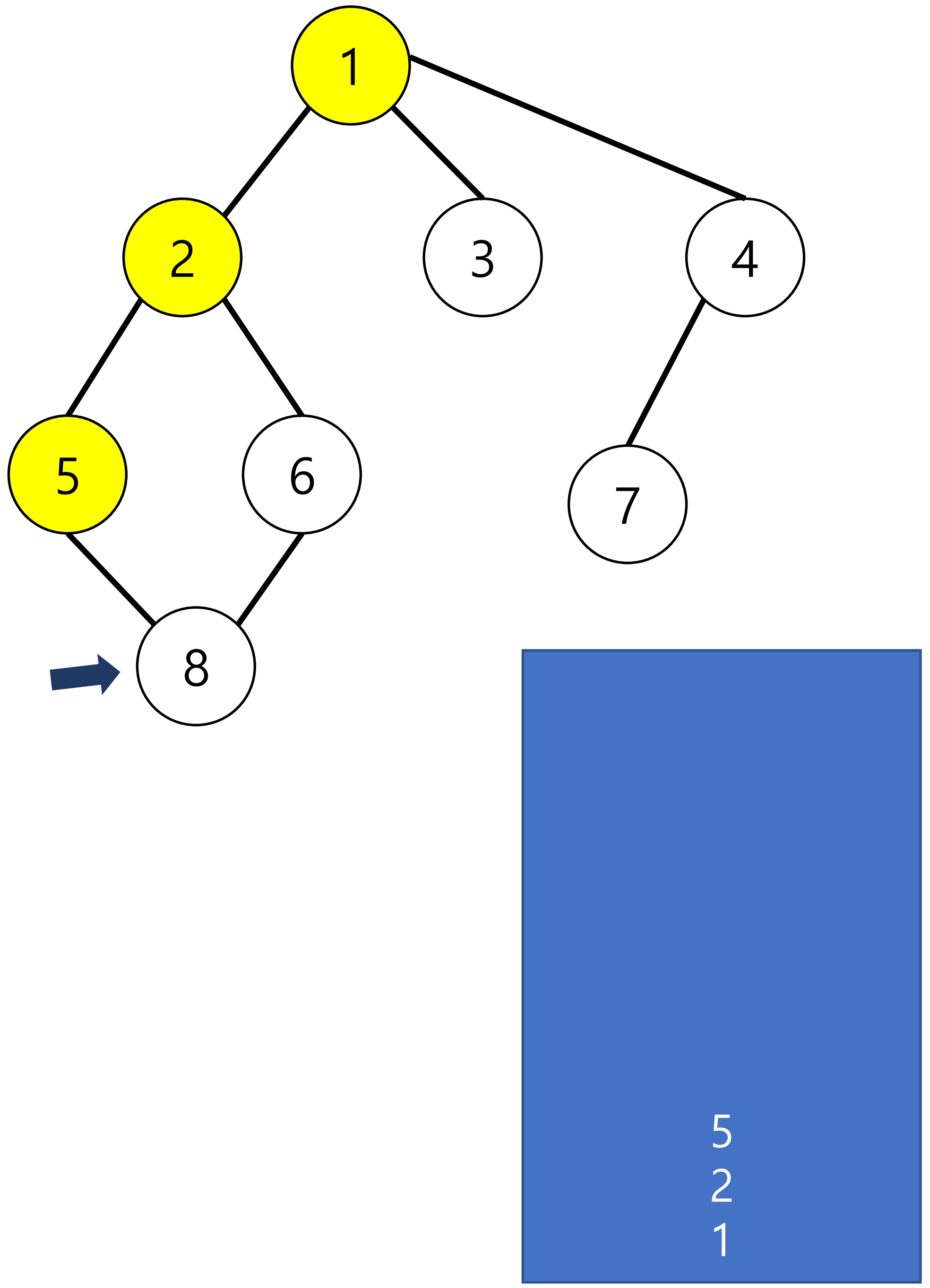


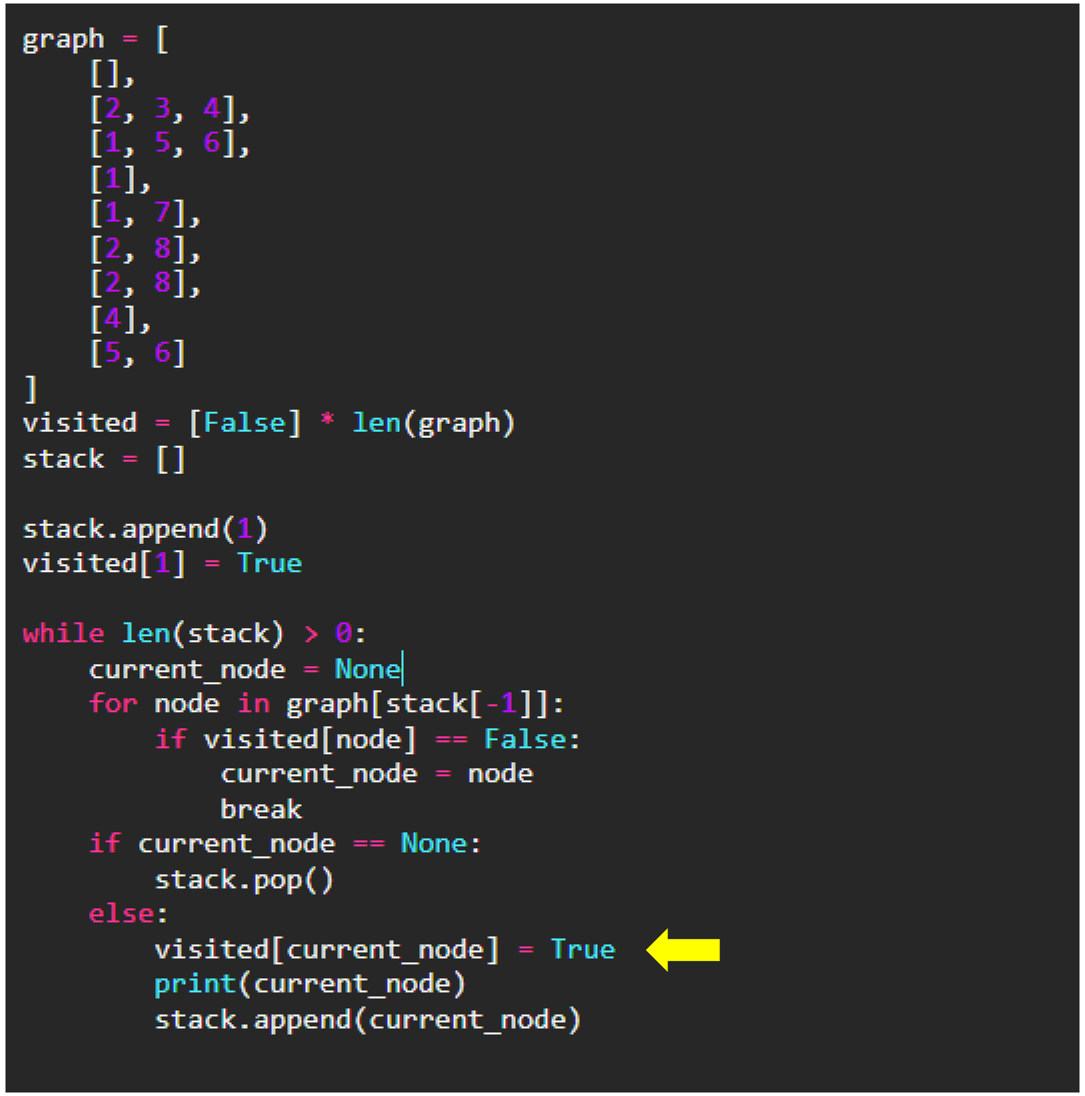
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]
visited = [False] * len(graph)
stack = []

stack.append(1)
visited[1] = True

while len(stack) > 0:
    current_node = None
    for node in graph[stack[-1]]:
        if visited[node] == False:
            current_node = node
            break
    if current_node == None:
        stack.pop()
    else:
        visited[current_node] = True
        print(current_node)
        stack.append(current_node)
```



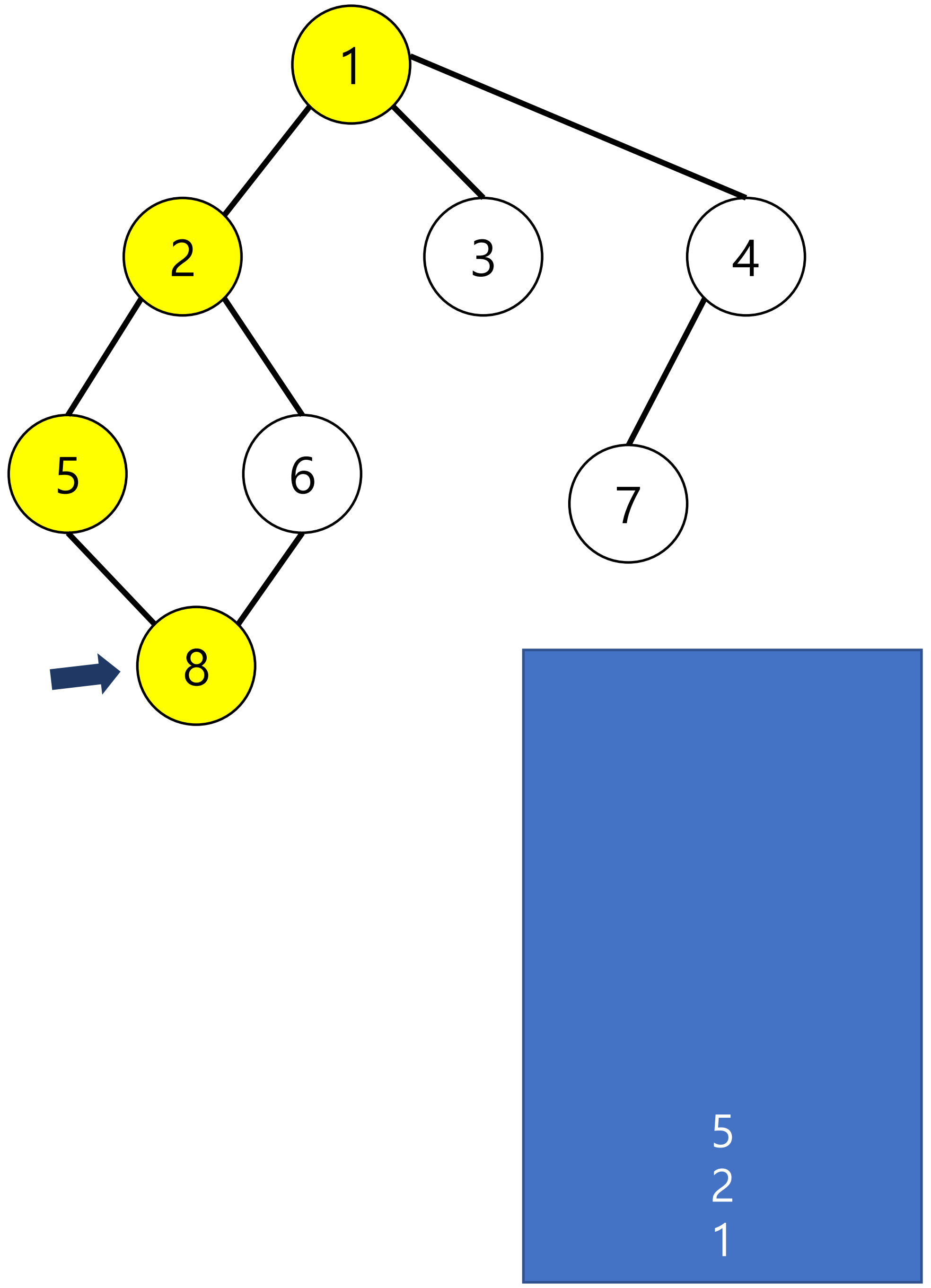


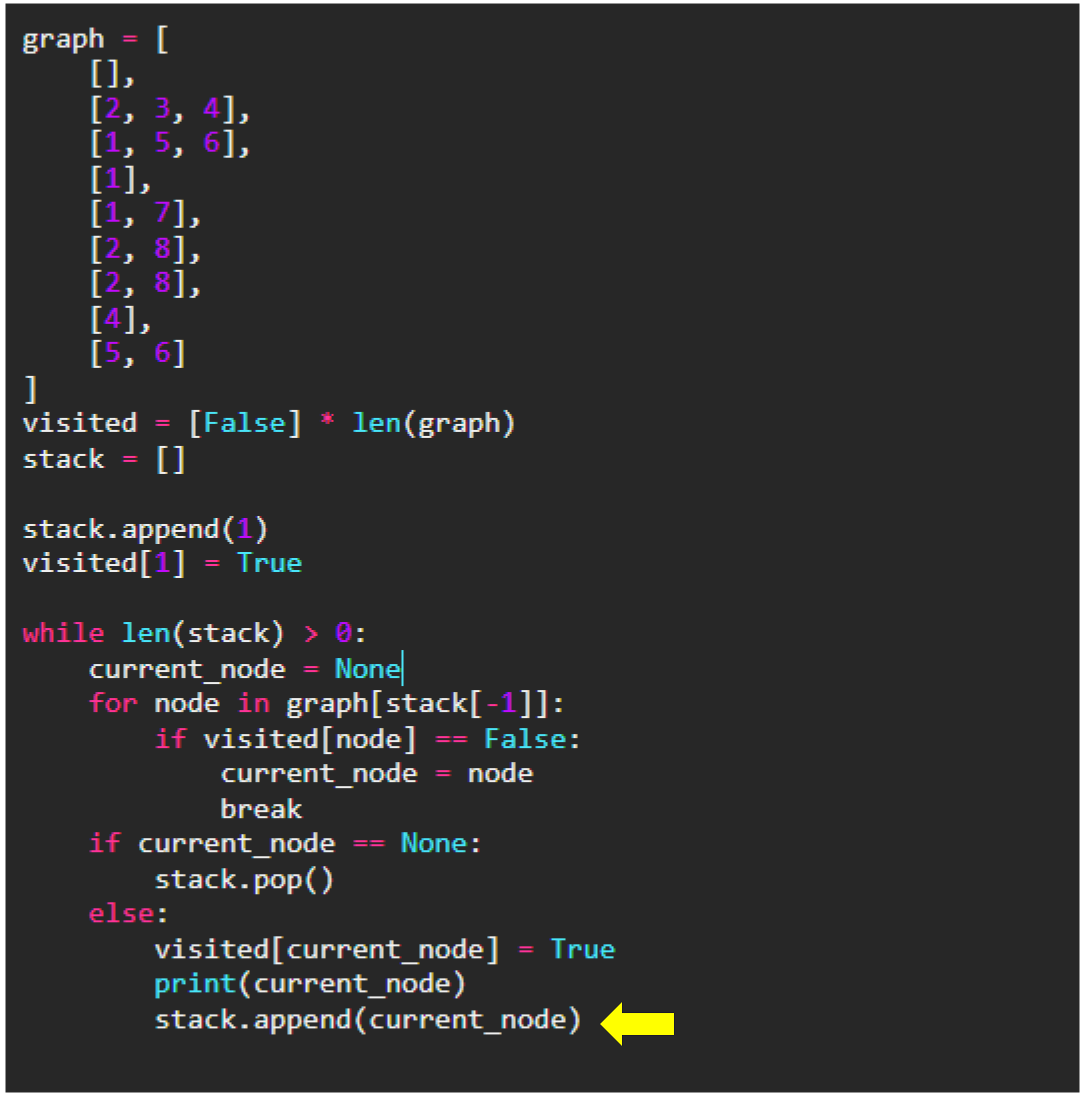
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]
visited = [False] * len(graph)
stack = []

stack.append(1)
visited[1] = True

while len(stack) > 0:
    current_node = None
    for node in graph[stack[-1]]:
        if visited[node] == False:
            current_node = node
            break
    if current_node == None:
        stack.pop()
    else:
        visited[current_node] = True
        print(current_node)
        stack.append(current_node)
```



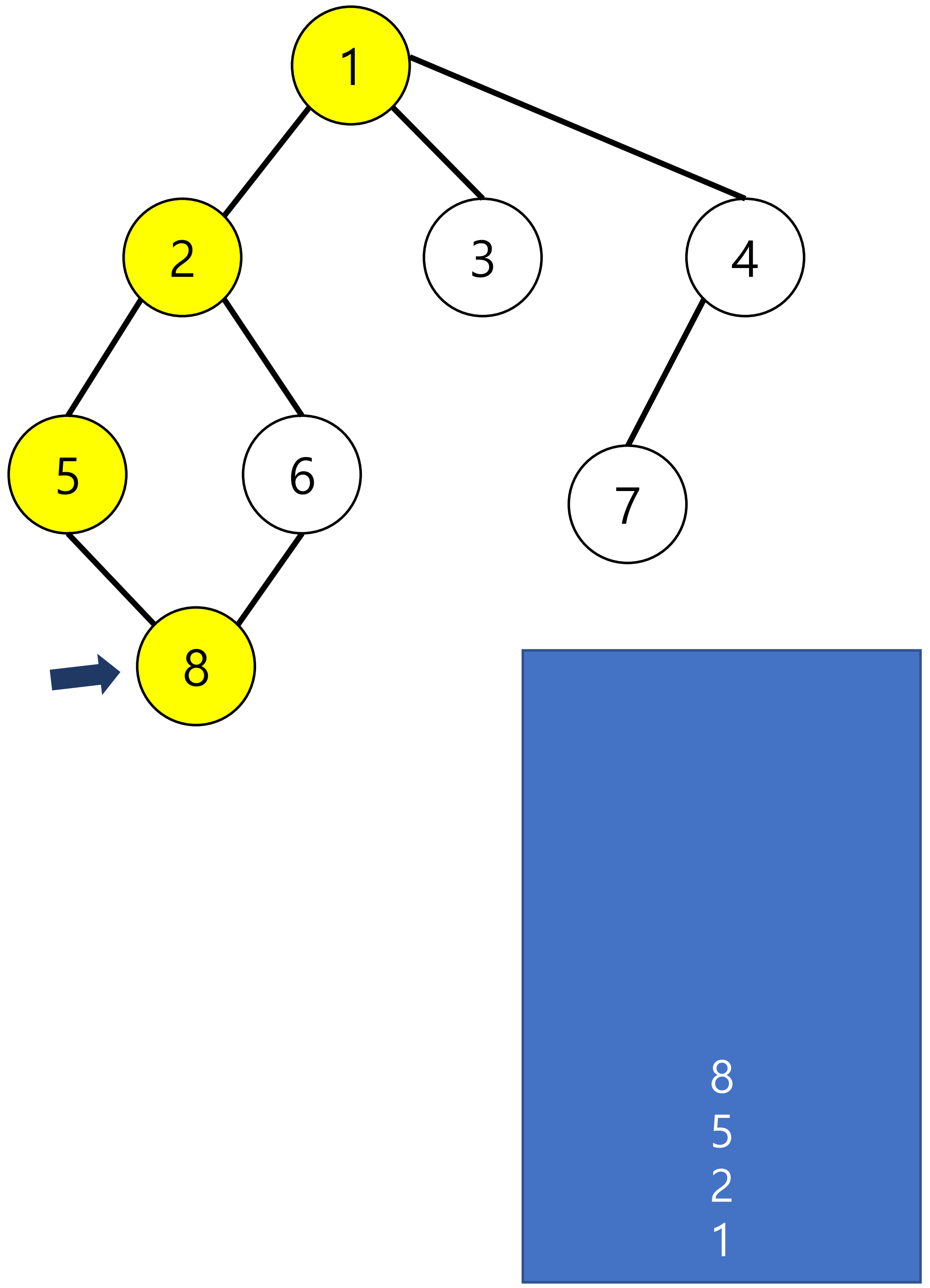


```python
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]
visited = [False] * len(graph)
stack = []

stack.append(1)
visited[1] = True

while len(stack) > 0:
    current_node = None
    for node in graph[stack[-1]]:
        if visited[node] == False:
            current_node = node
            break
    if current_node == None:
        stack.pop()
    else:
        visited[current_node] = True
        print(current_node)
        stack.append(current_node)
```

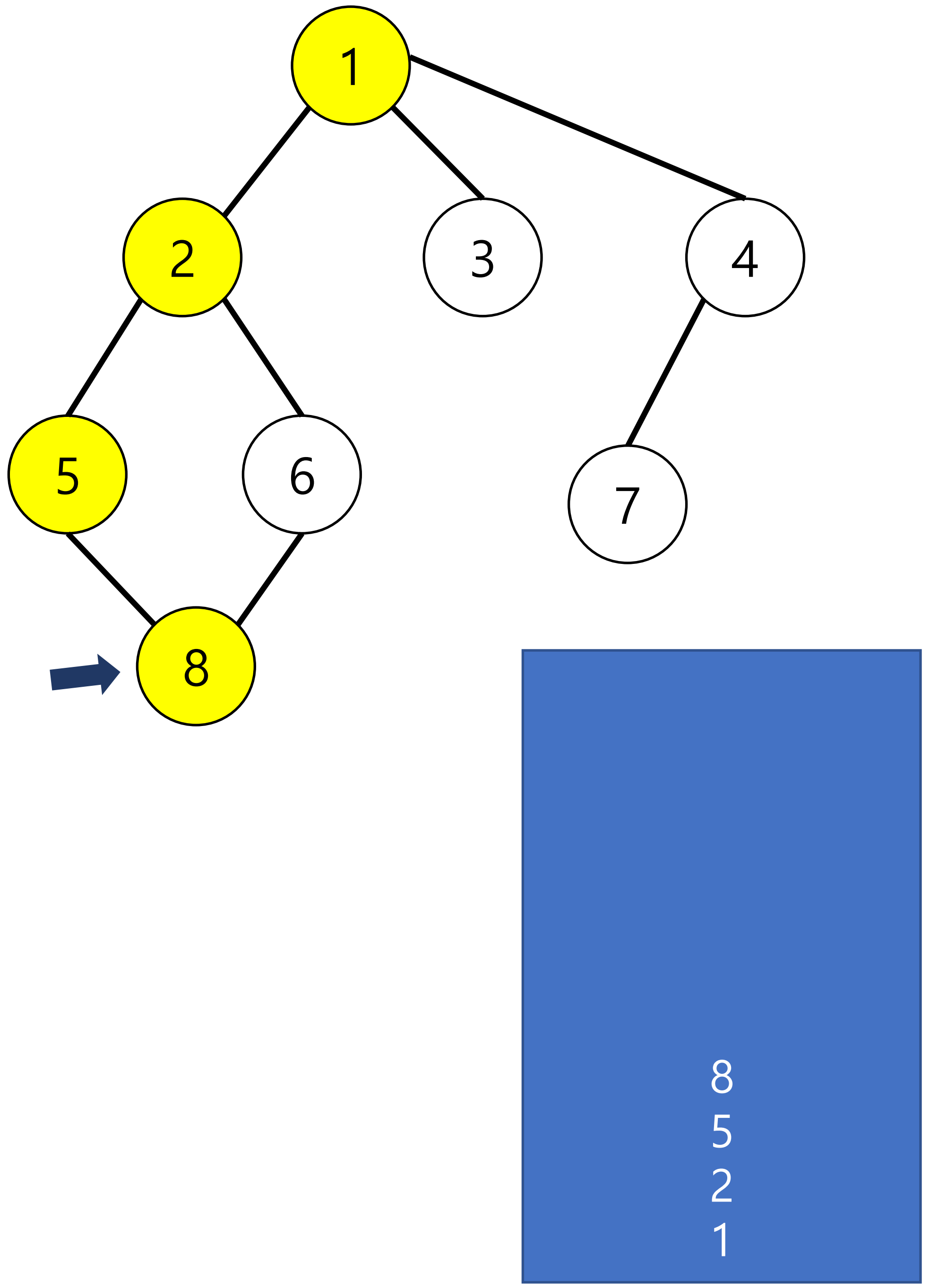


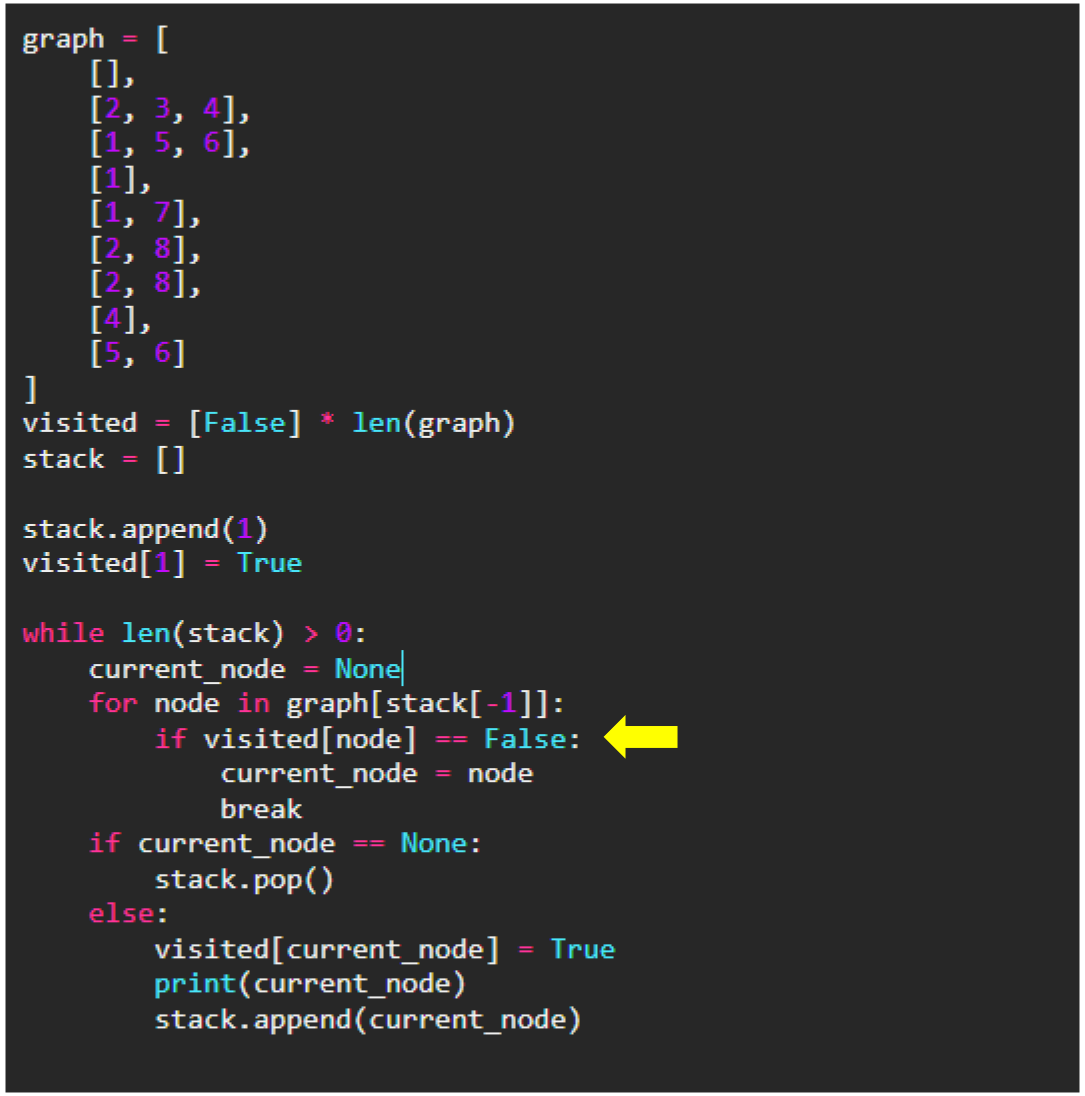


```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]
visited = [False] * len(graph)
stack = []

stack.append(1)
visited[1] = True

while len(stack) > 0:
    current_node = None
    for node in graph[stack[-1]]:
        if visited[node] == False:
            current_node = node
            break
    if current_node == None:
        stack.pop()
    else:
        visited[current_node] = True
        print(current_node)
        stack.append(current_node)
```

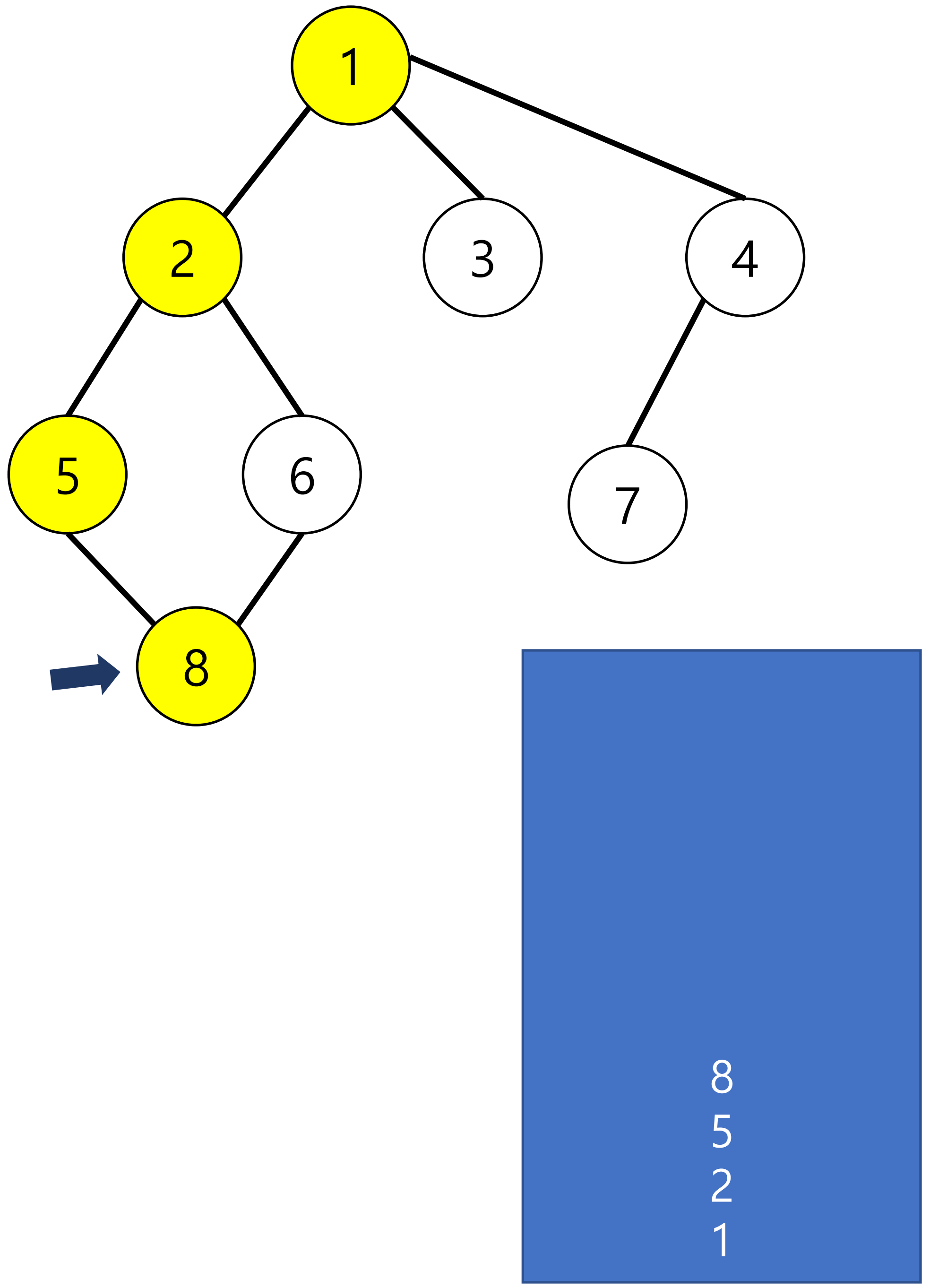


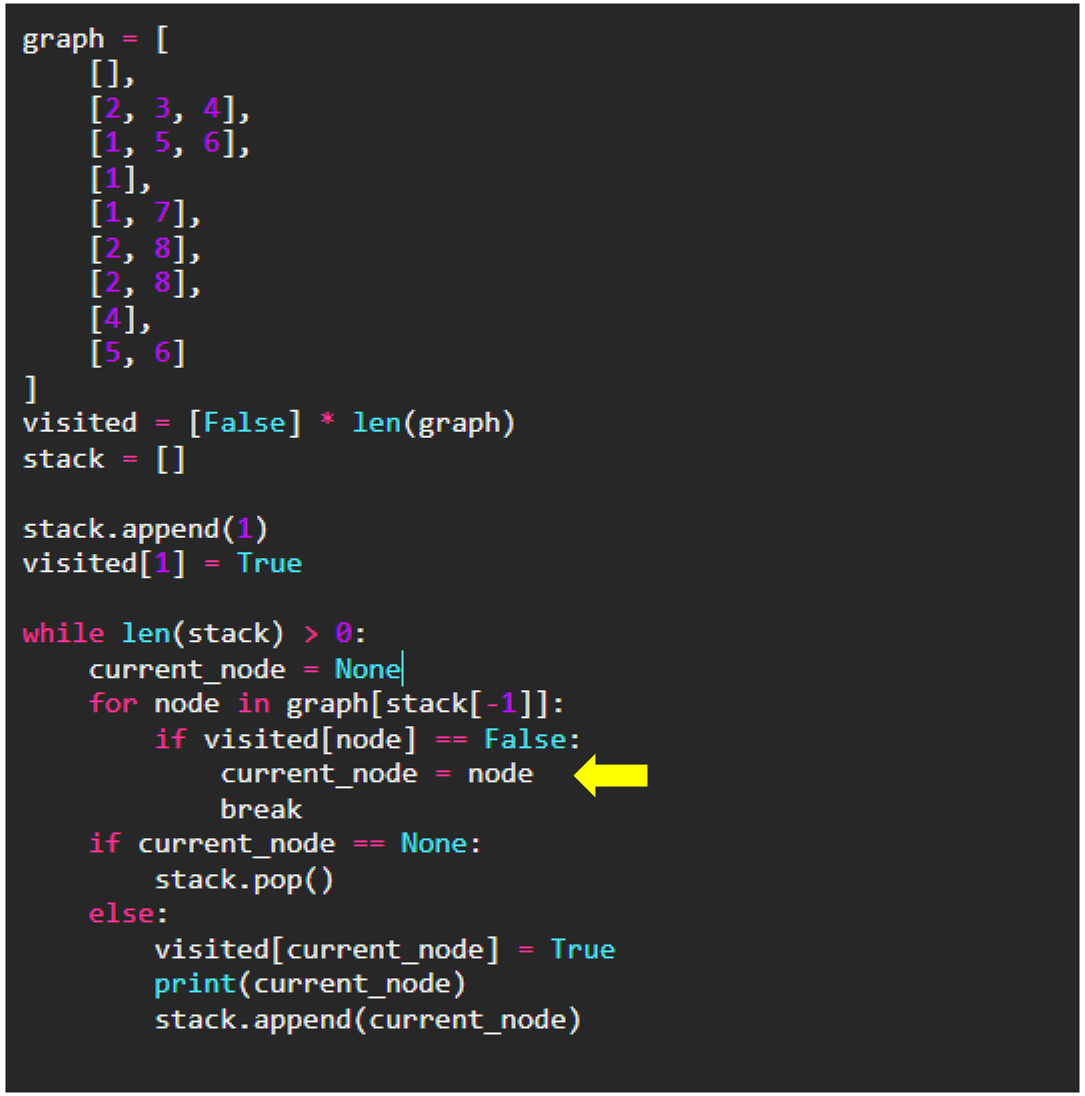
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]
visited = [False] * len(graph)
stack = []

stack.append(1)
visited[1] = True

while len(stack) > 0:
    current_node = None
    for node in graph[stack[-1]]:
        if visited[node] == False:
            current_node = node
            break
    if current_node == None:
        stack.pop()
    else:
        visited[current_node] = True
        print(current_node)
        stack.append(current_node)
```



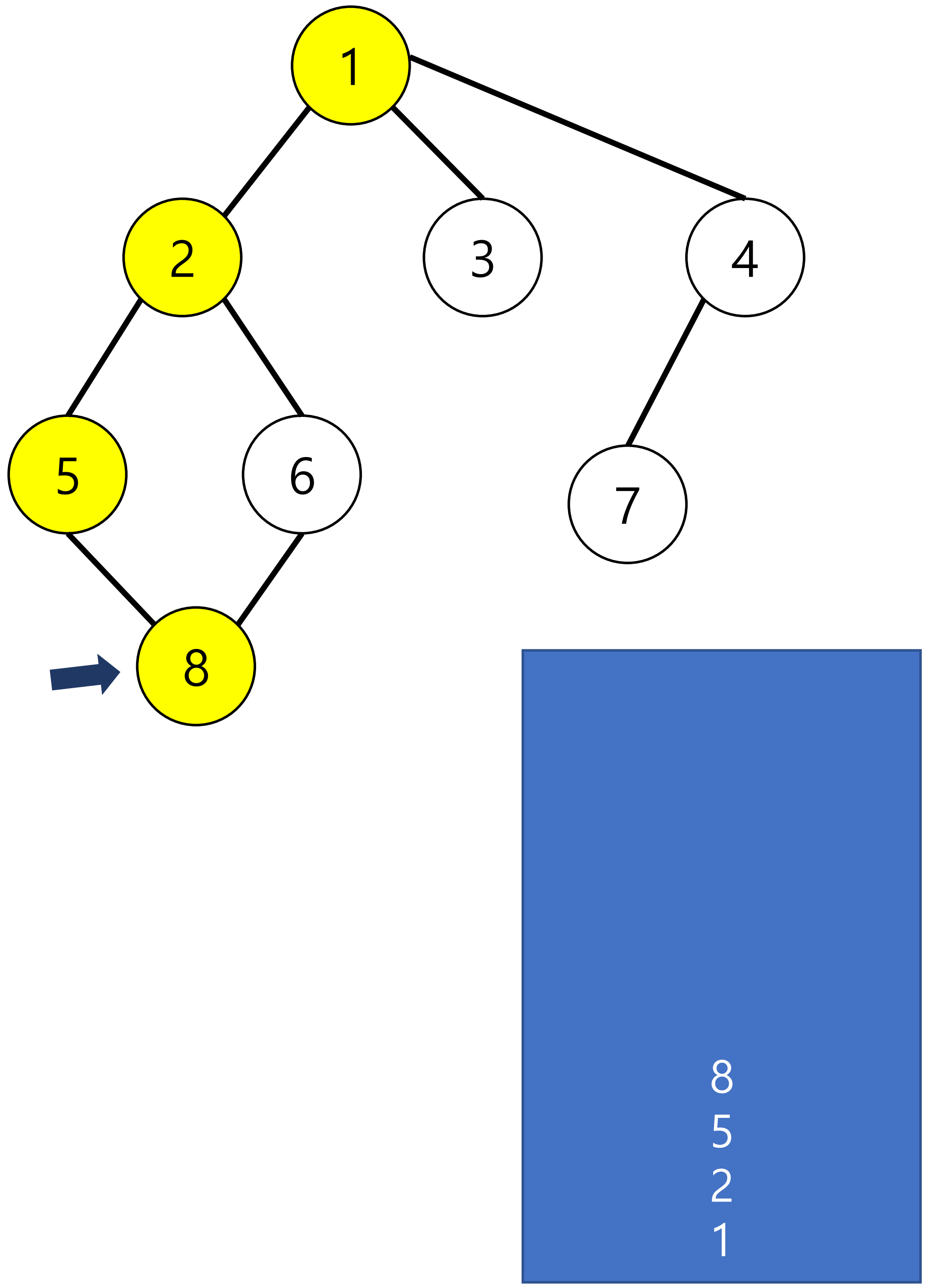


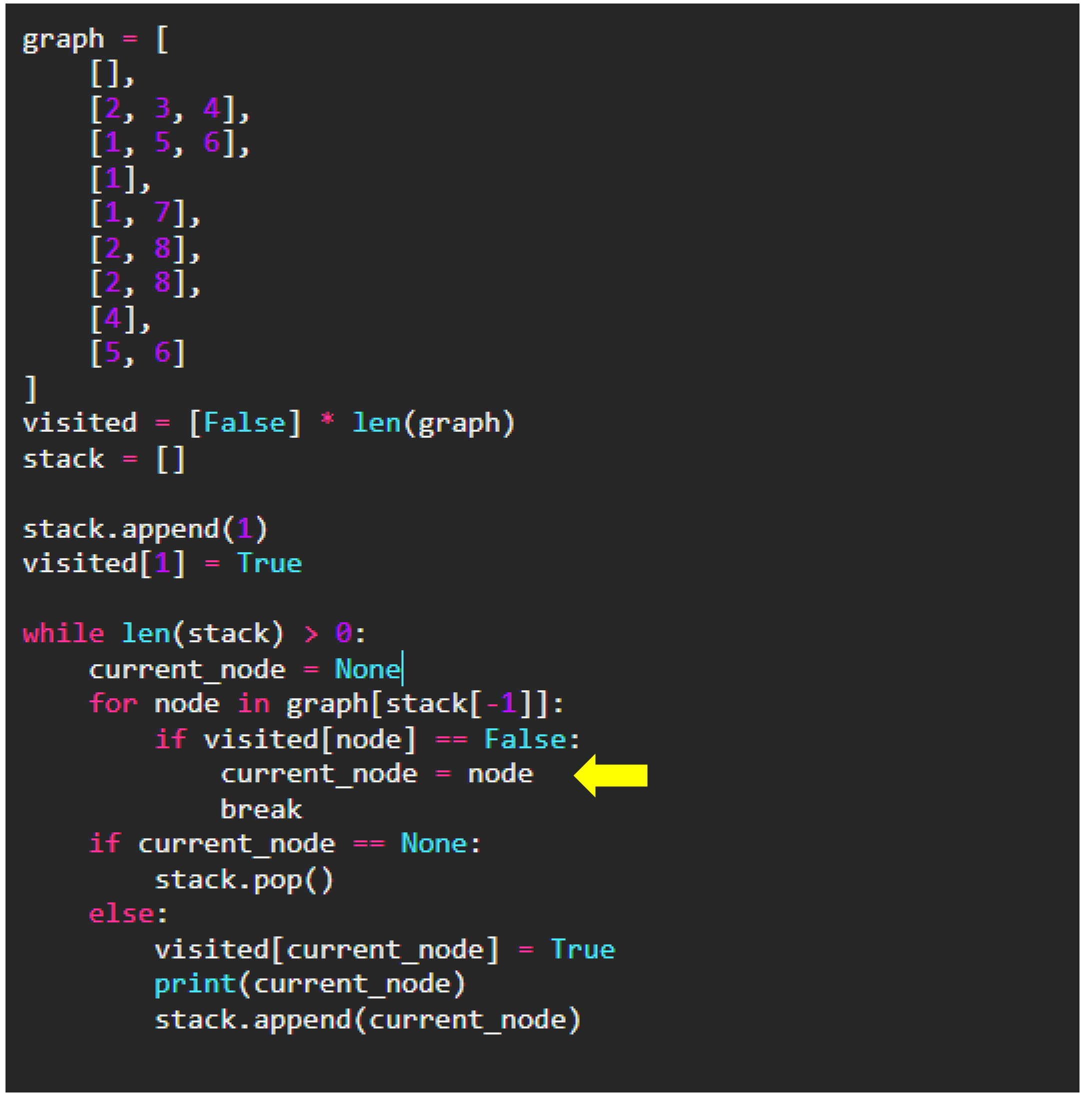
```python
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]
visited = [False] * len(graph)
stack = []

stack.append(1)
visited[1] = True

while len(stack) > 0:
    current_node = None
    for node in graph[stack[-1]]:
        if visited[node] == False:
            current_node = node
            break
    if current_node == None:
        stack.pop()
    else:
        visited[current_node] = True
        print(current_node)
        stack.append(current_node)
```



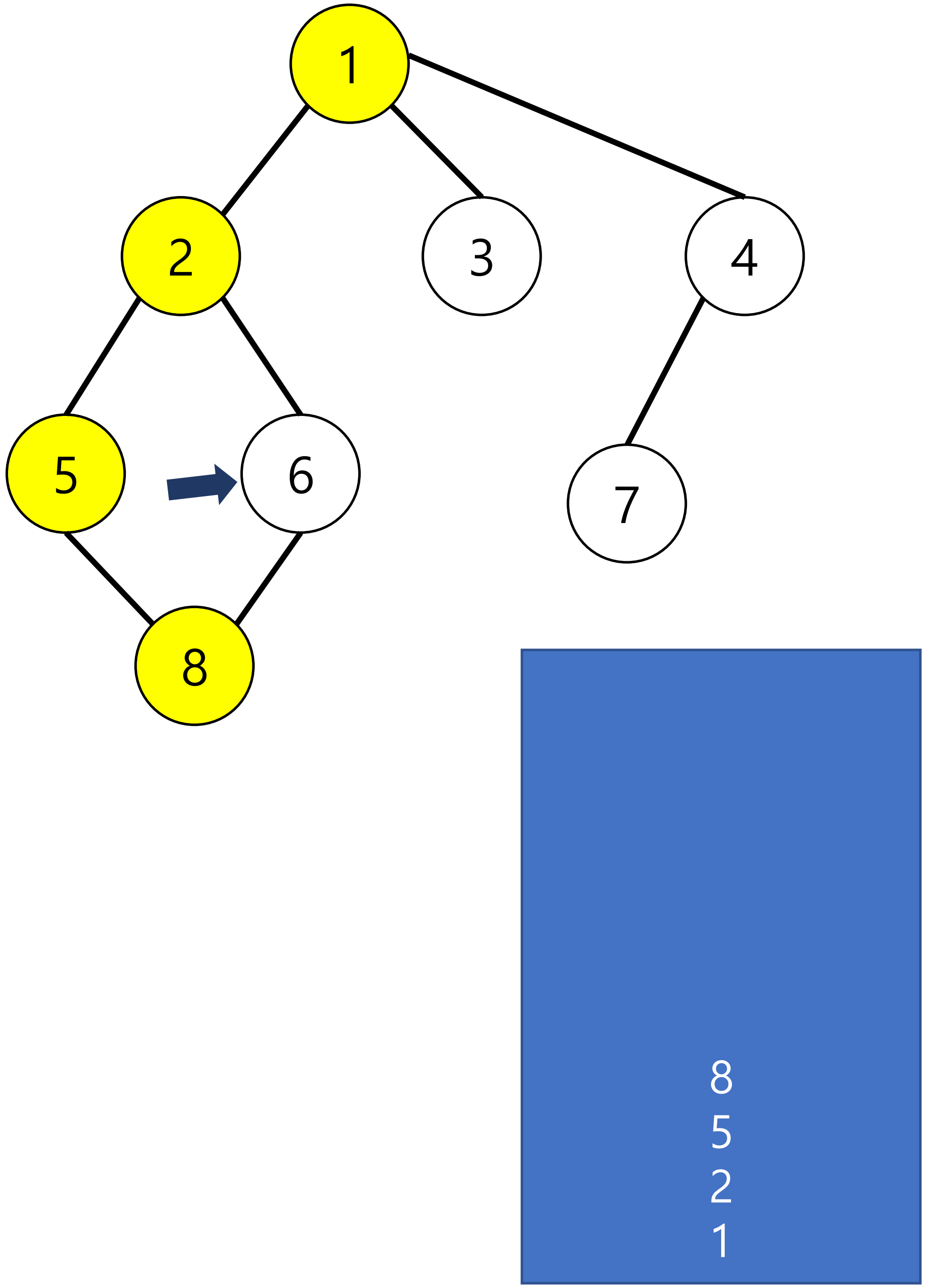


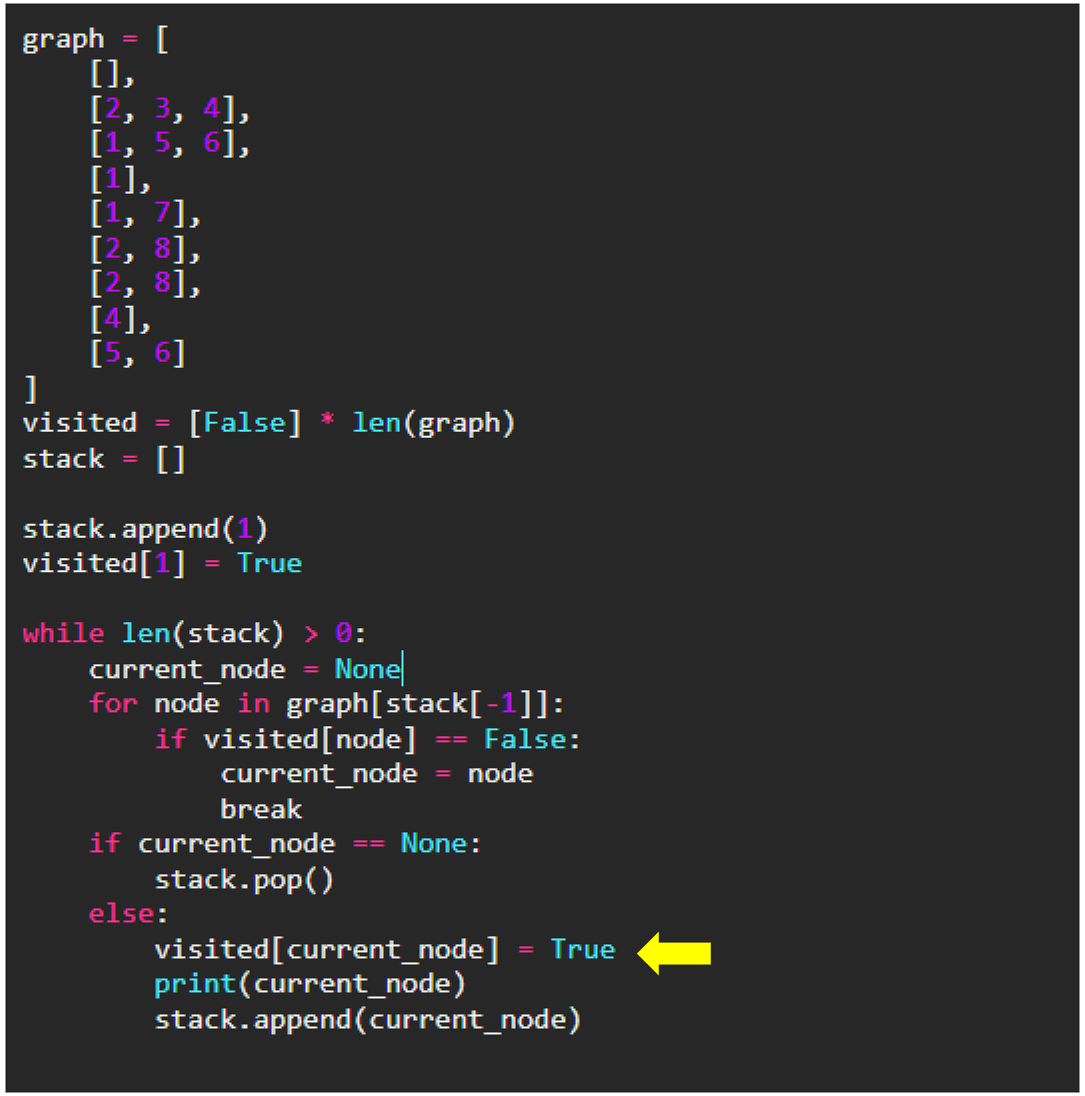
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]
visited = [False] * len(graph)
stack = []

stack.append(1)
visited[1] = True

while len(stack) > 0:
    current_node = None
    for node in graph[stack[-1]]:
        if visited[node] == False:
            current_node = node
            break
    if current_node == None:
        stack.pop()
    else:
        visited[current_node] = True
        print(current_node)
        stack.append(current_node)
```



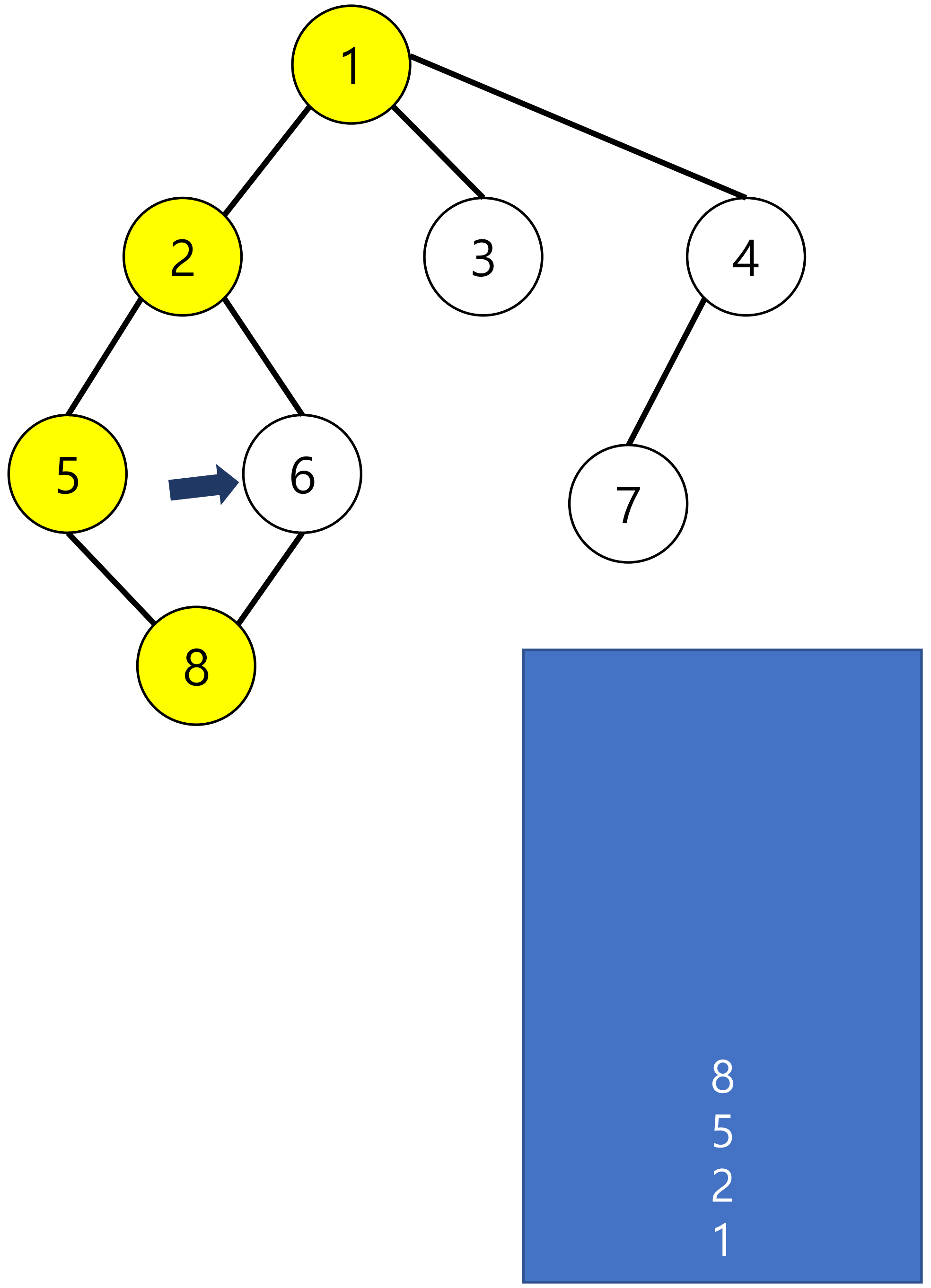


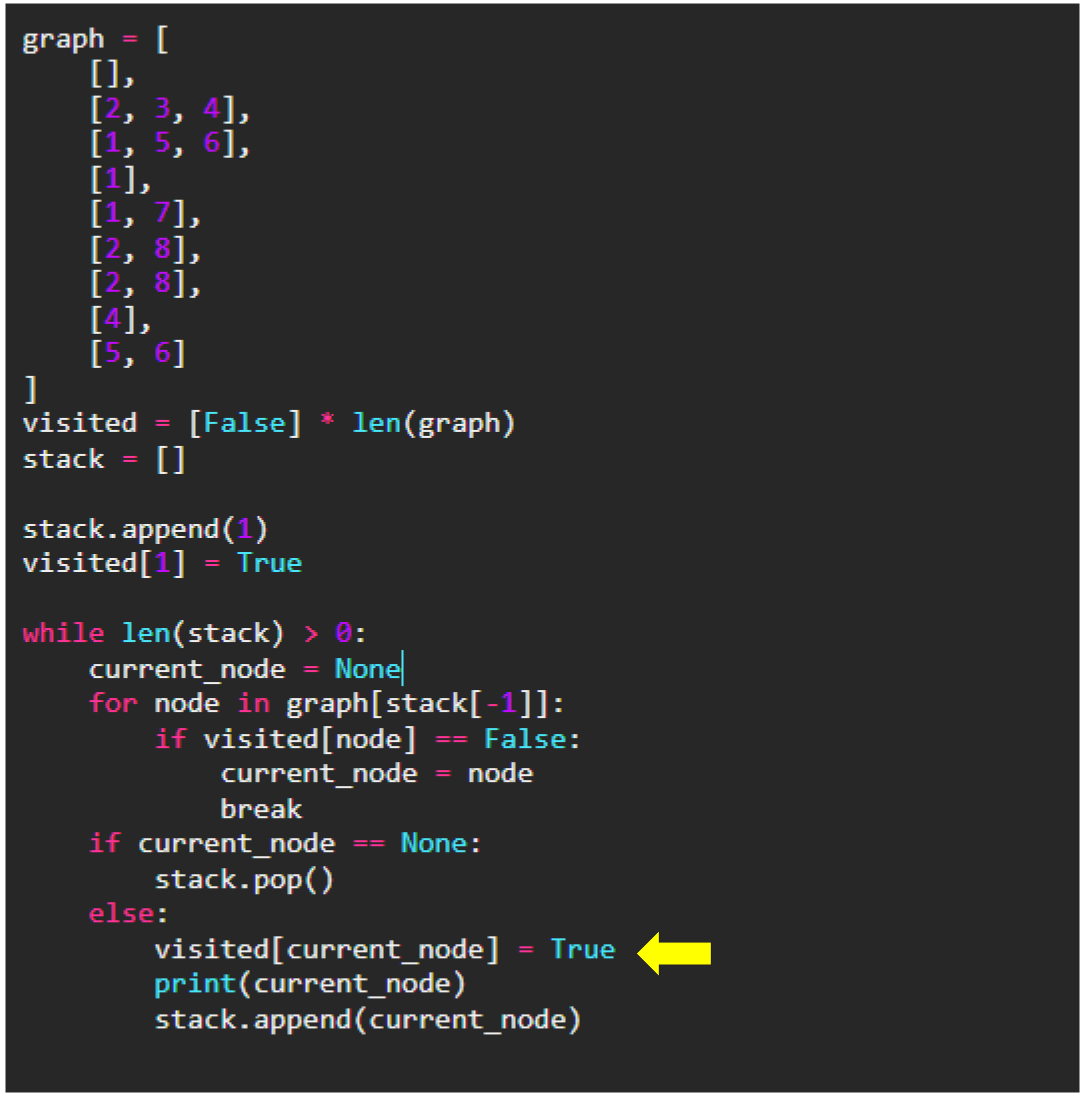
```python
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]
visited = [False] * len(graph)
stack = []

stack.append(1)
visited[1] = True

while len(stack) > 0:
    current_node = None
    for node in graph[stack[-1]]:
        if visited[node] == False:
            current_node = node
            break
    if current_node == None:
        stack.pop()
    else:
        visited[current_node] = True
        print(current_node)
        stack.append(current_node)
```



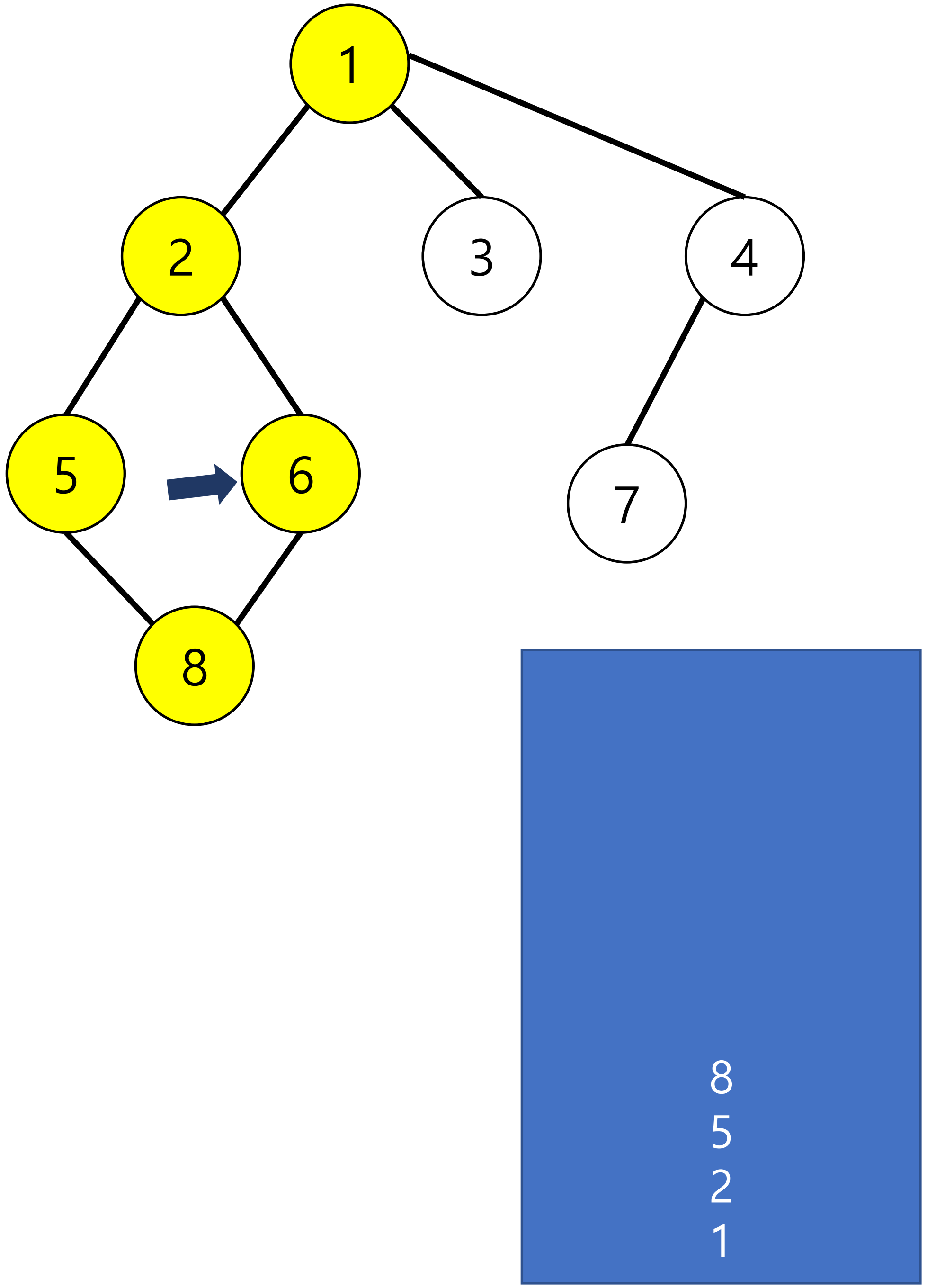


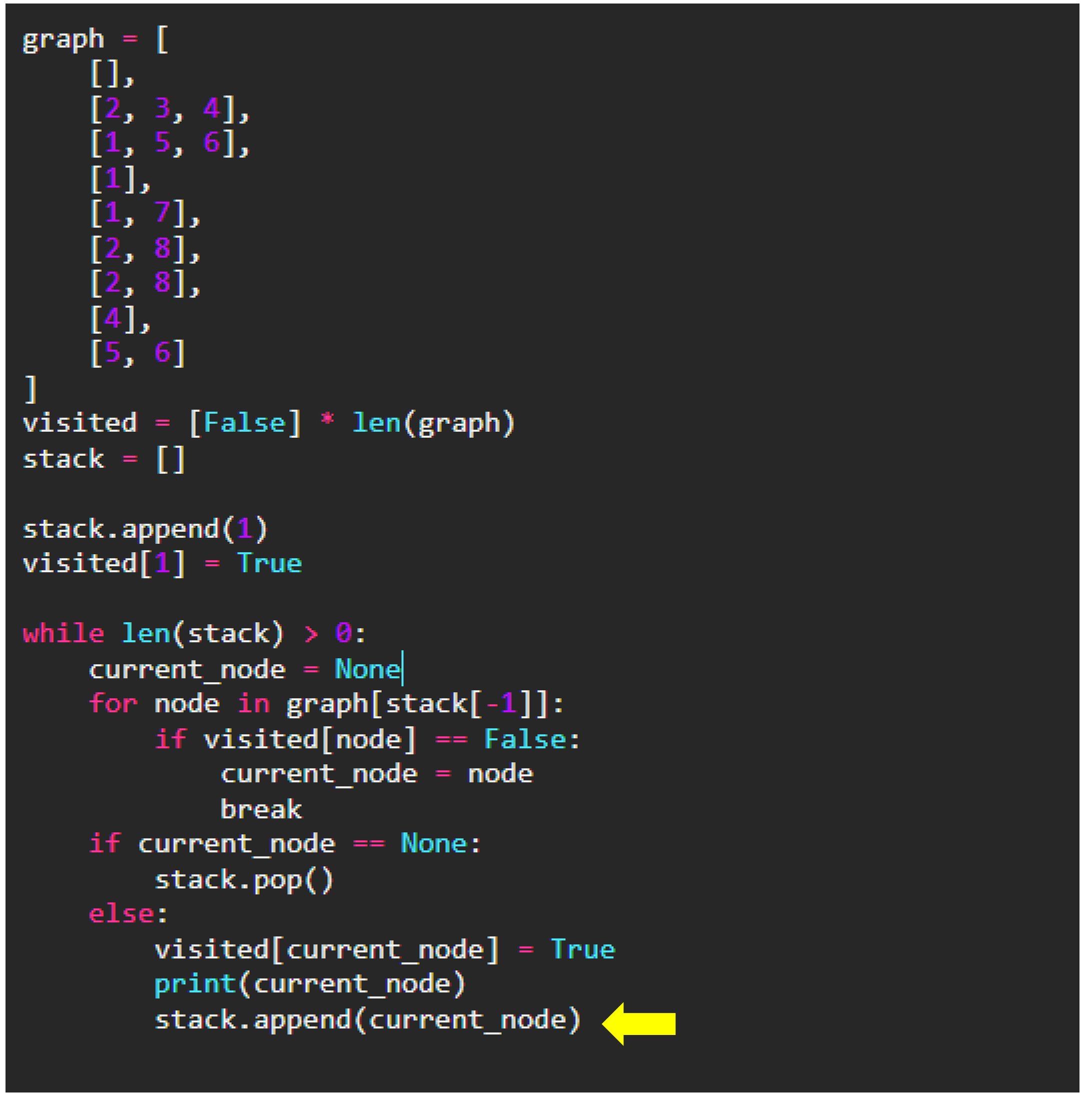
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]
visited = [False] * len(graph)
stack = []

stack.append(1)
visited[1] = True

while len(stack) > 0:
    current_node = None
    for node in graph[stack[-1]]:
        if visited[node] == False:
            current_node = node
            break
    if current_node == None:
        stack.pop()
    else:
        visited[current_node] = True
        print(current_node)
        stack.append(current_node)
```



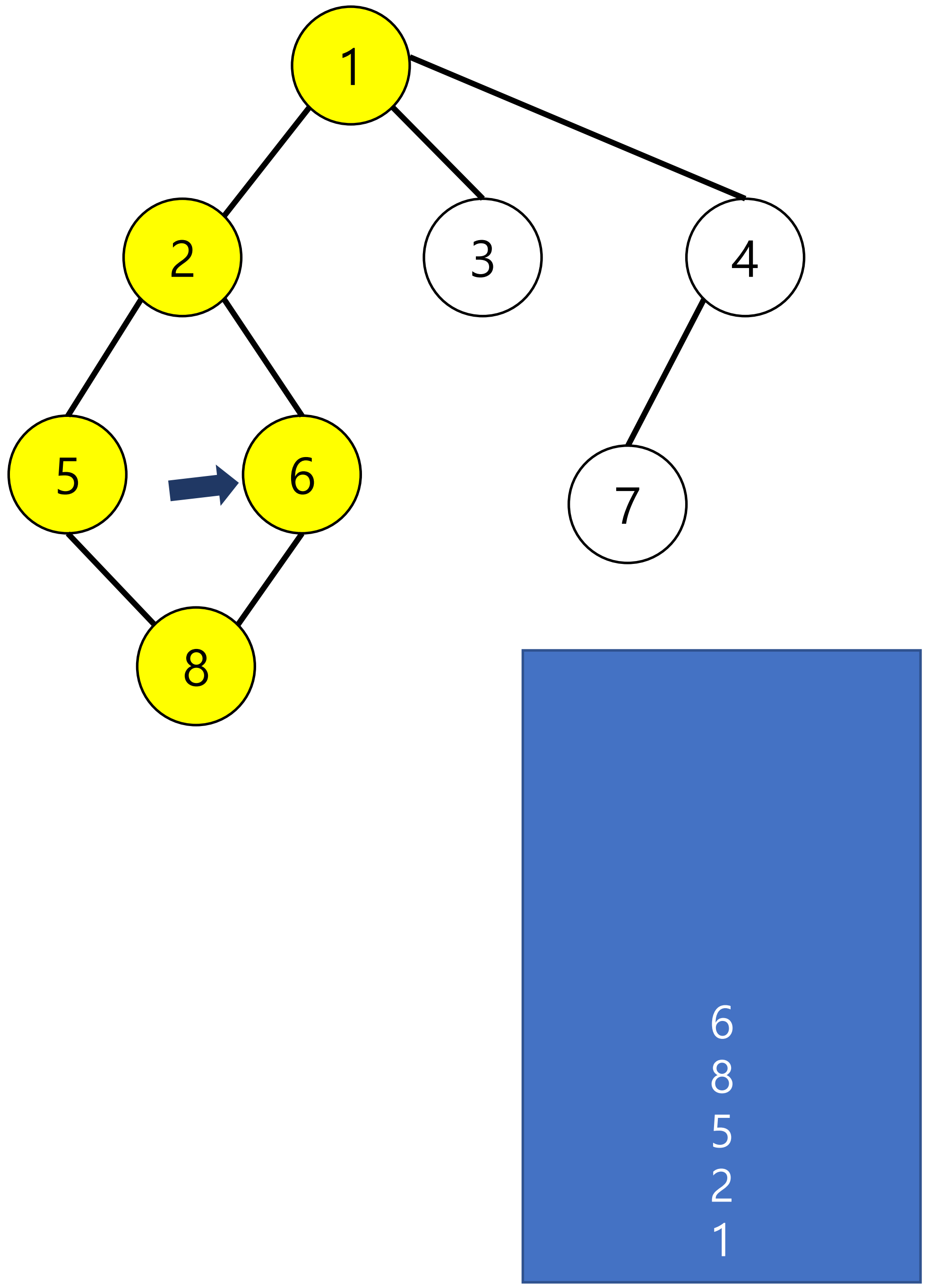


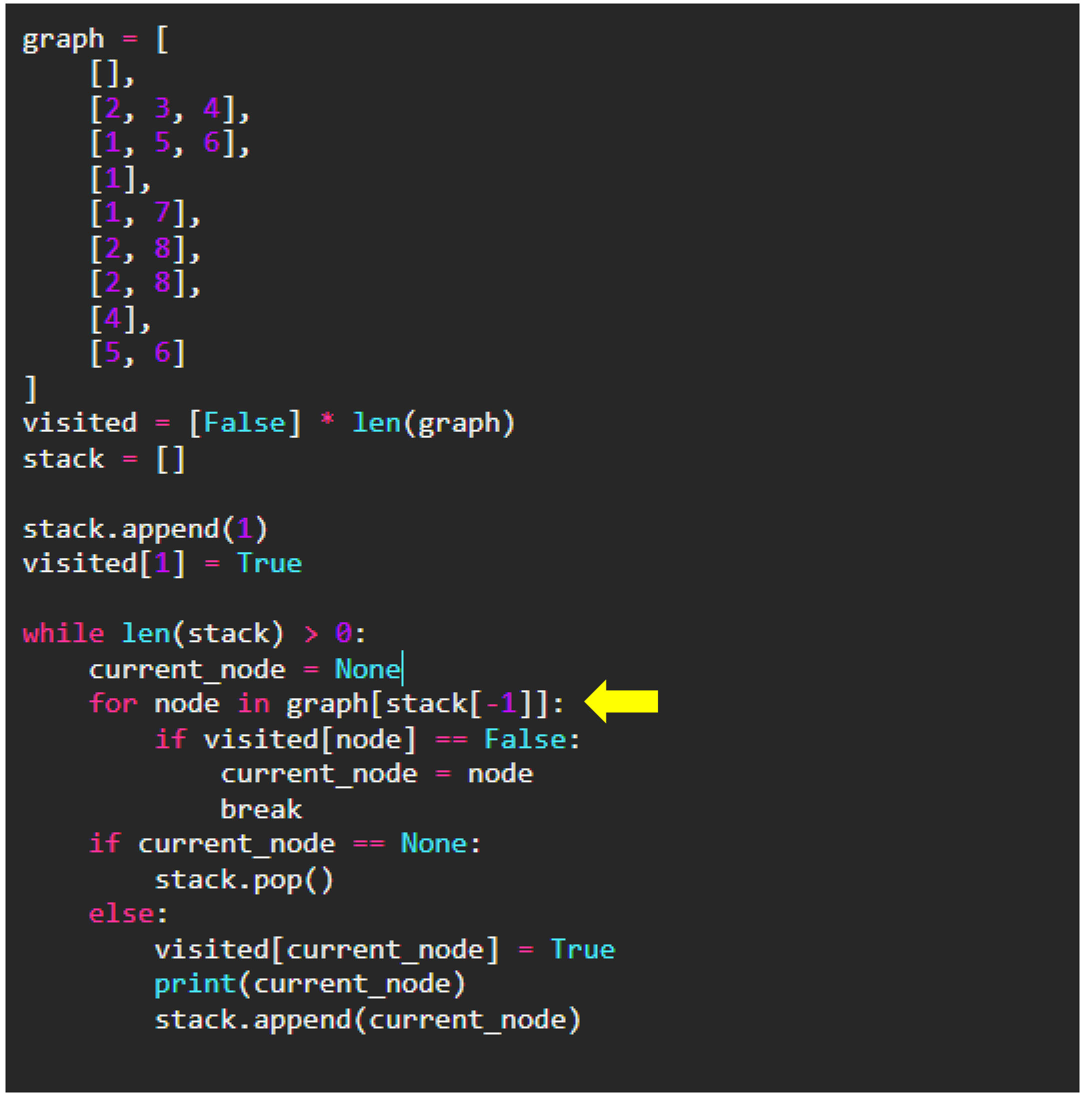
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]
visited = [False] * len(graph)
stack = []

stack.append(1)
visited[1] = True

while len(stack) > 0:
    current_node = None
    for node in graph[stack[-1]]:
        if visited[node] == False:
            current_node = node
            break
    if current_node == None:
        stack.pop()
    else:
        visited[current_node] = True
        print(current_node)
        stack.append(current_node)
```



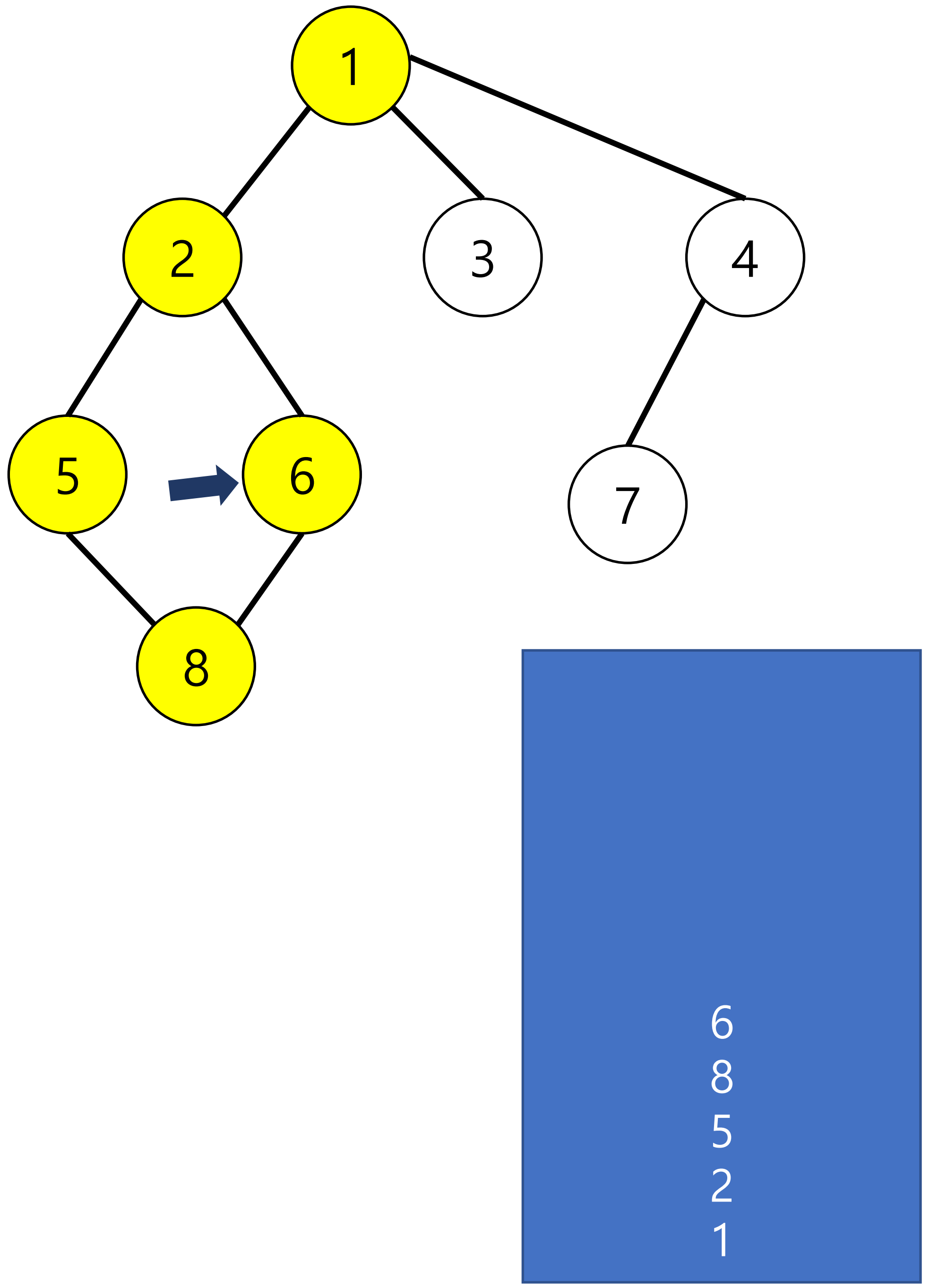


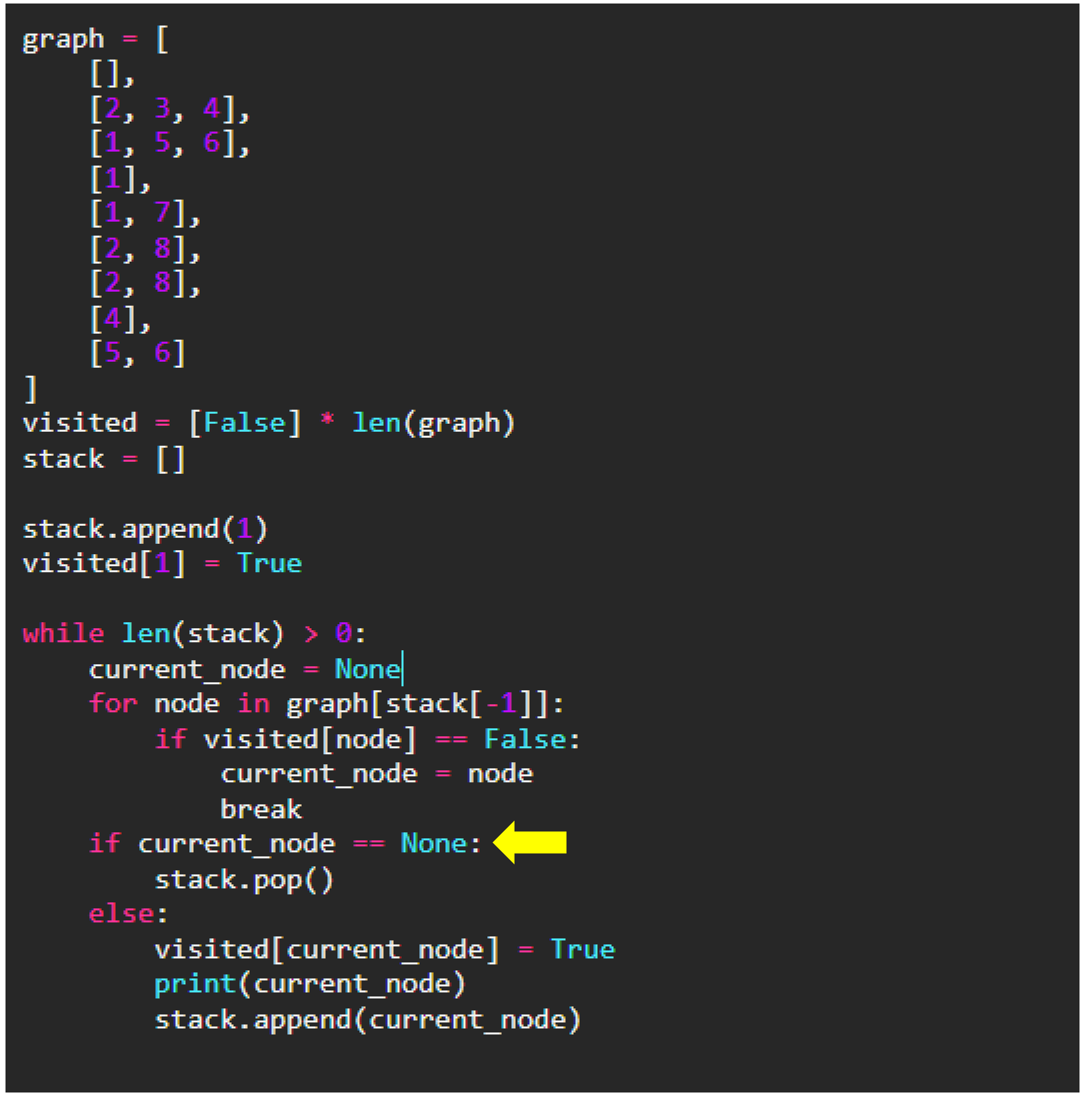
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]
visited = [False] * len(graph)
stack = []

stack.append(1)
visited[1] = True

while len(stack) > 0:
    current_node = None
    for node in graph[stack[-1]]:
        if visited[node] == False:
            current_node = node
            break
    if current_node == None:
        stack.pop()
    else:
        visited[current_node] = True
        print(current_node)
        stack.append(current_node)
```



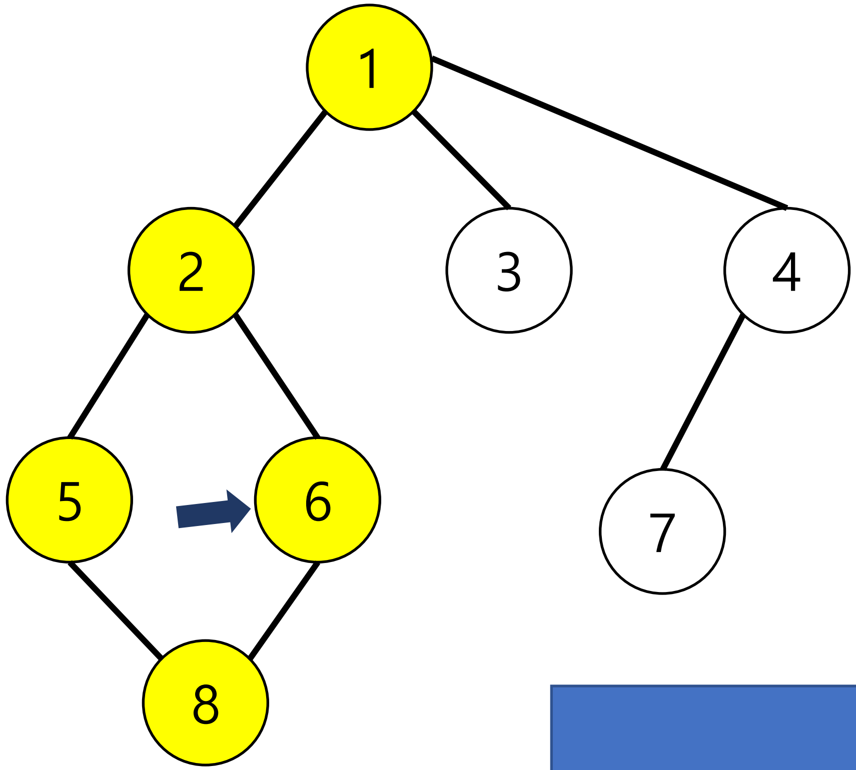


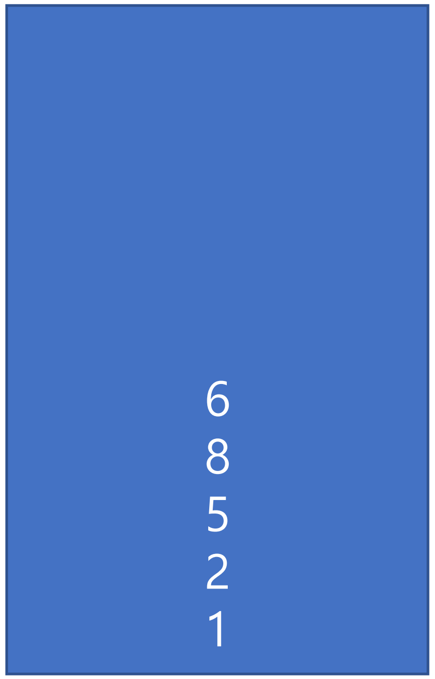
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]
visited = [False] * len(graph)
stack = []

stack.append(1)
visited[1] = True

while len(stack) > 0:
    current_node = None
    for node in graph[stack[-1]]:
        if visited[node] == False:
            current_node = node
            break
    if current_node == None:
        stack.pop()
    else:
        visited[current_node] = True
        print(current_node)
        stack.append(current_node)
```

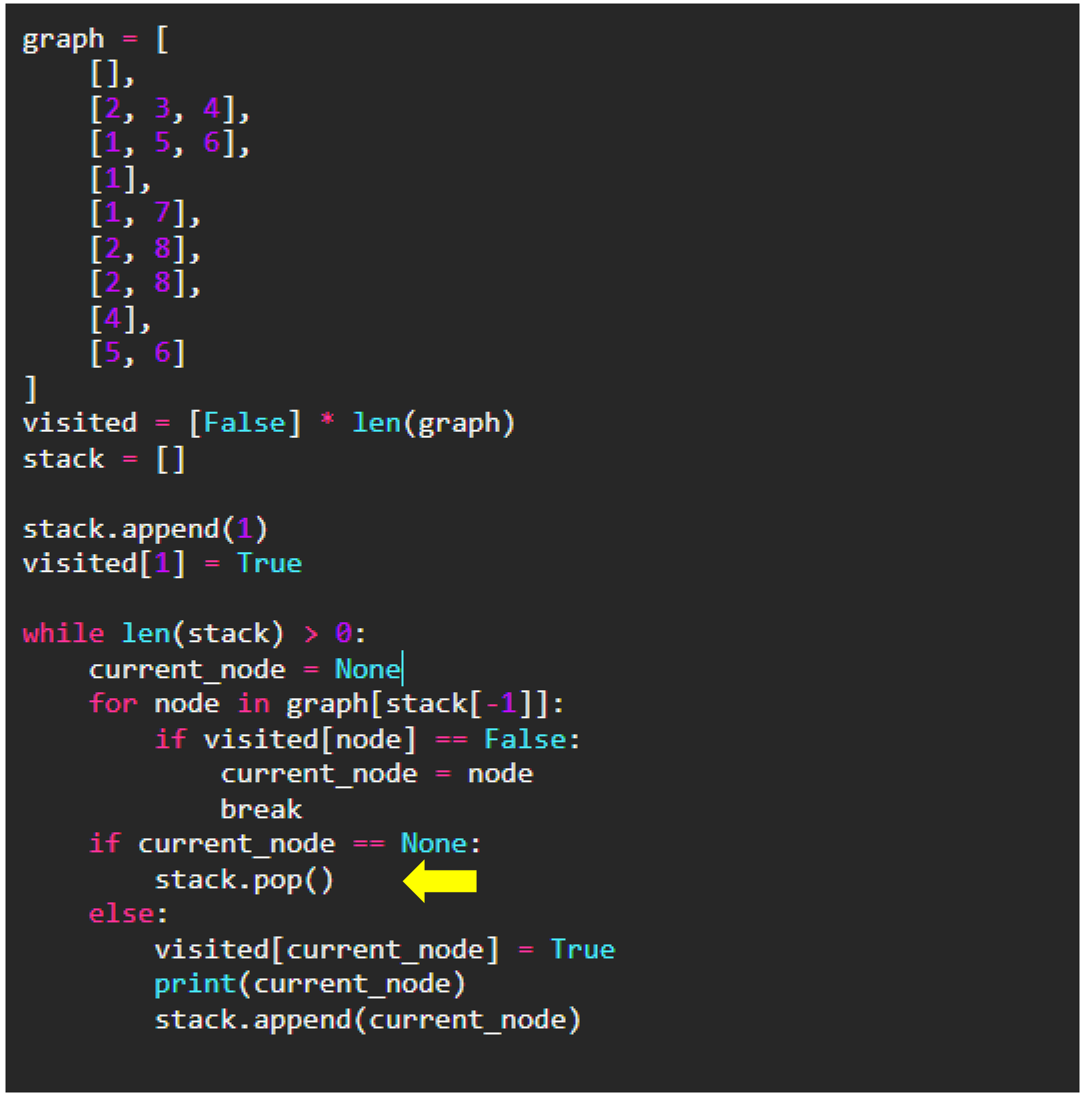




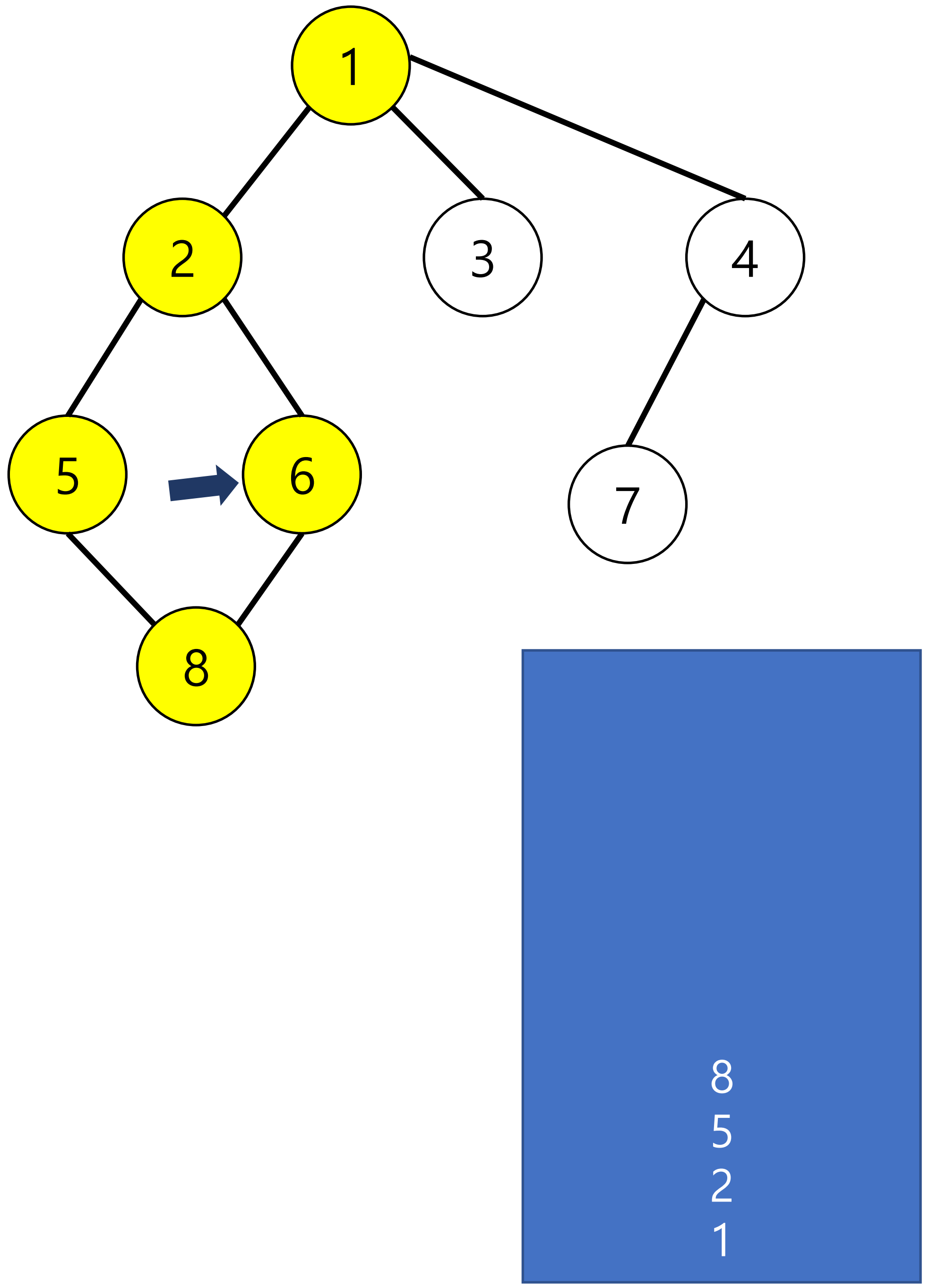
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]
visited = [False] * len(graph)
stack = []

stack.append(1)
visited[1] = True

while len(stack) > 0:
    current_node = None
    for node in graph[stack[-1]]:
        if visited[node] == False:
            current_node = node
            break
    if current_node == None:
        stack.pop()
    else:
        visited[current_node] = True
        print(current_node)
        stack.append(current_node)
```

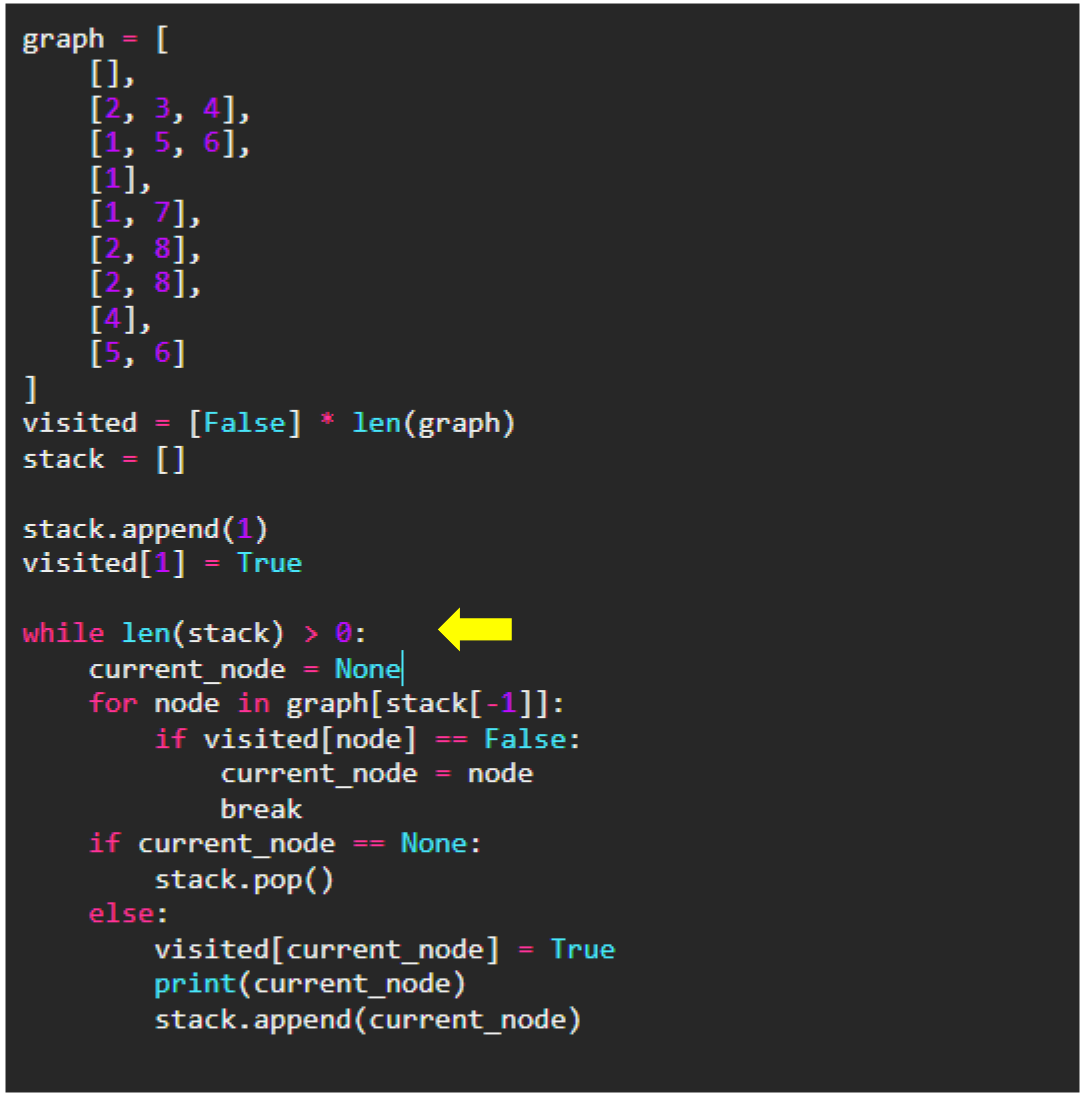




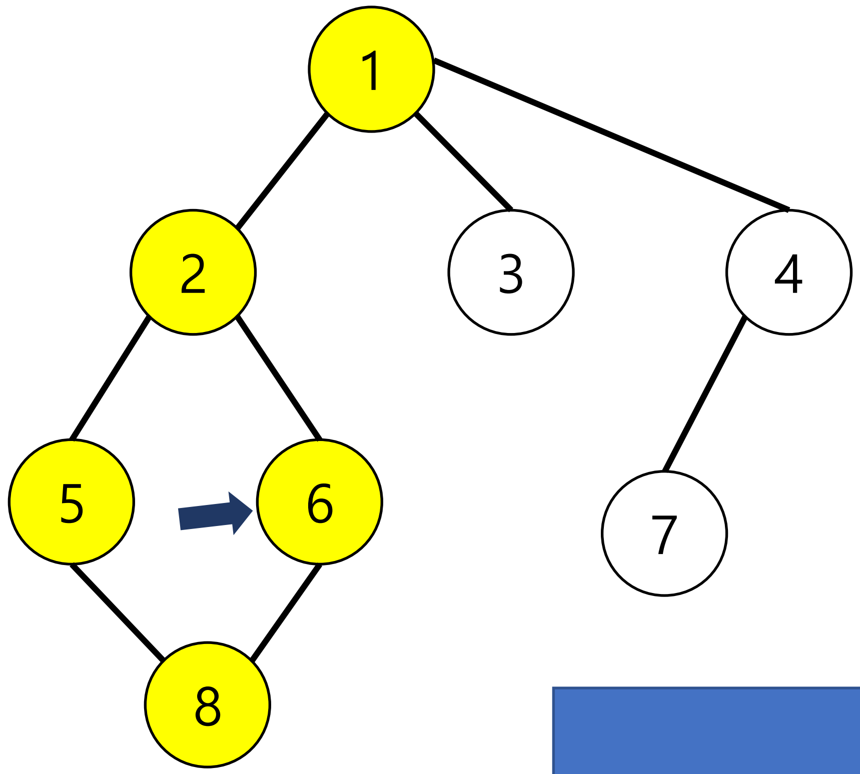


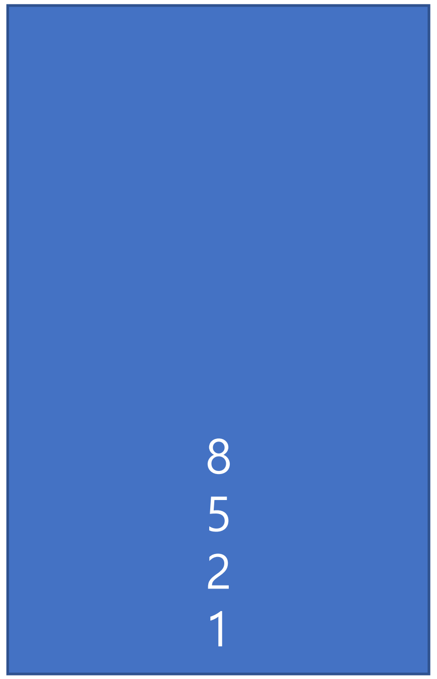
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]
visited = [False] * len(graph)
stack = []

stack.append(1)
visited[1] = True

while len(stack) > 0:
    current_node = None
    for node in graph[stack[-1]]:
        if visited[node] == False:
            current_node = node
            break
    if current_node == None:
        stack.pop()
    else:
        visited[current_node] = True
        print(current_node)
        stack.append(current_node)
```

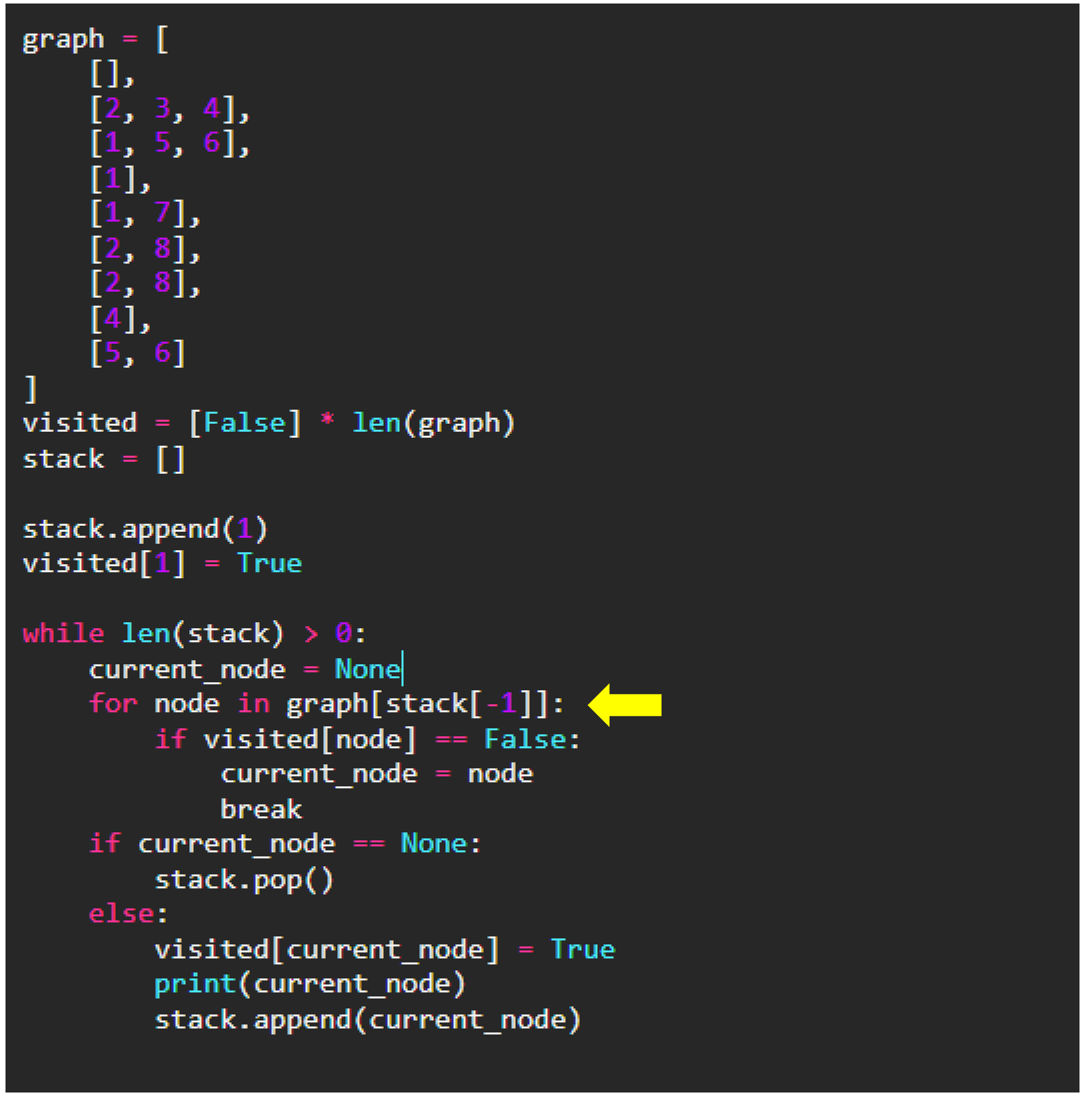




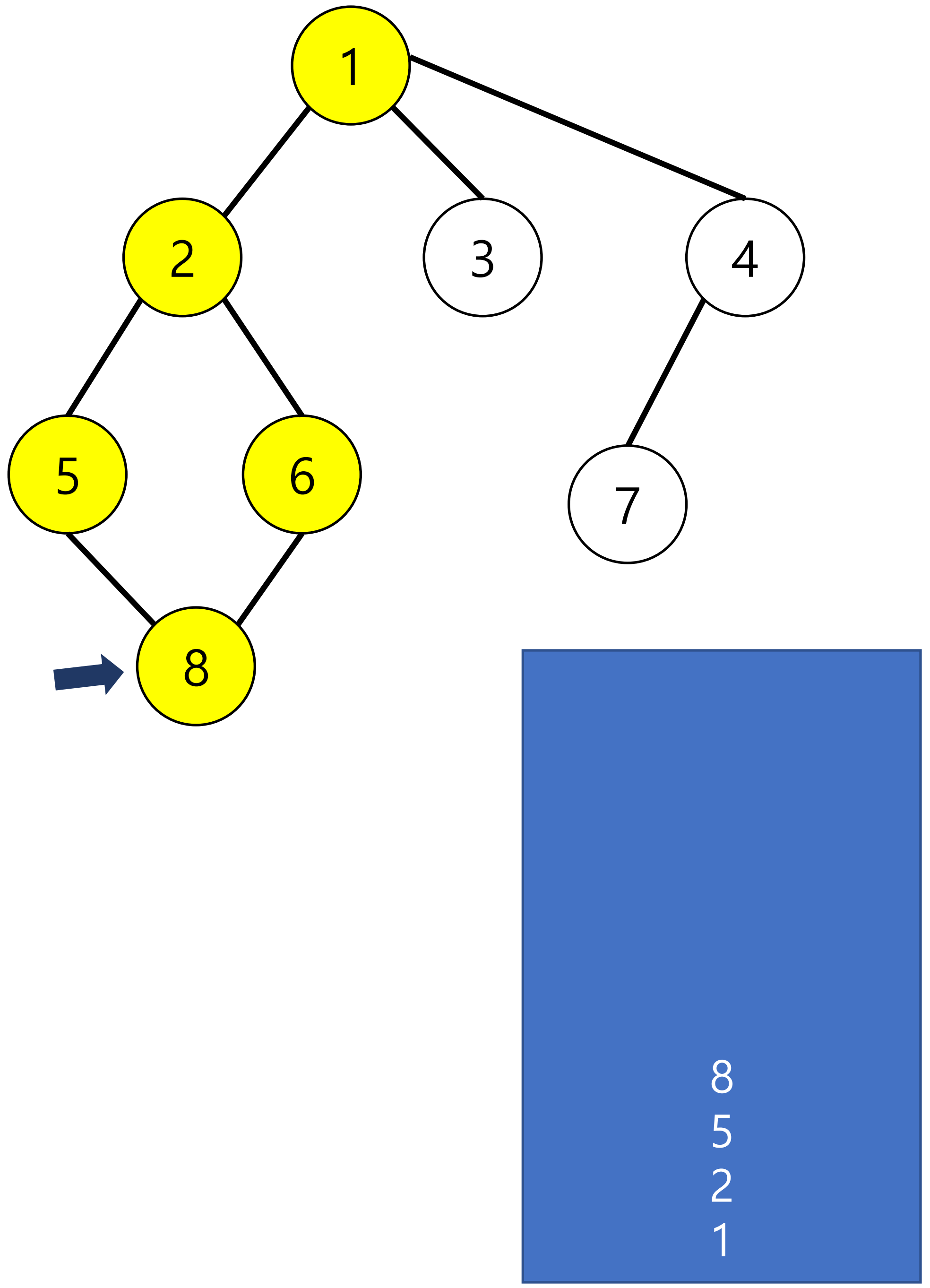
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]
visited = [False] * len(graph)
stack = []

stack.append(1)
visited[1] = True

while len(stack) > 0:
    current_node = None
    for node in graph[stack[-1]]:
        if visited[node] == False:
            current_node = node
            break
    if current_node == None:
        stack.pop()
    else:
        visited[current_node] = True
        print(current_node)
        stack.append(current_node)
```

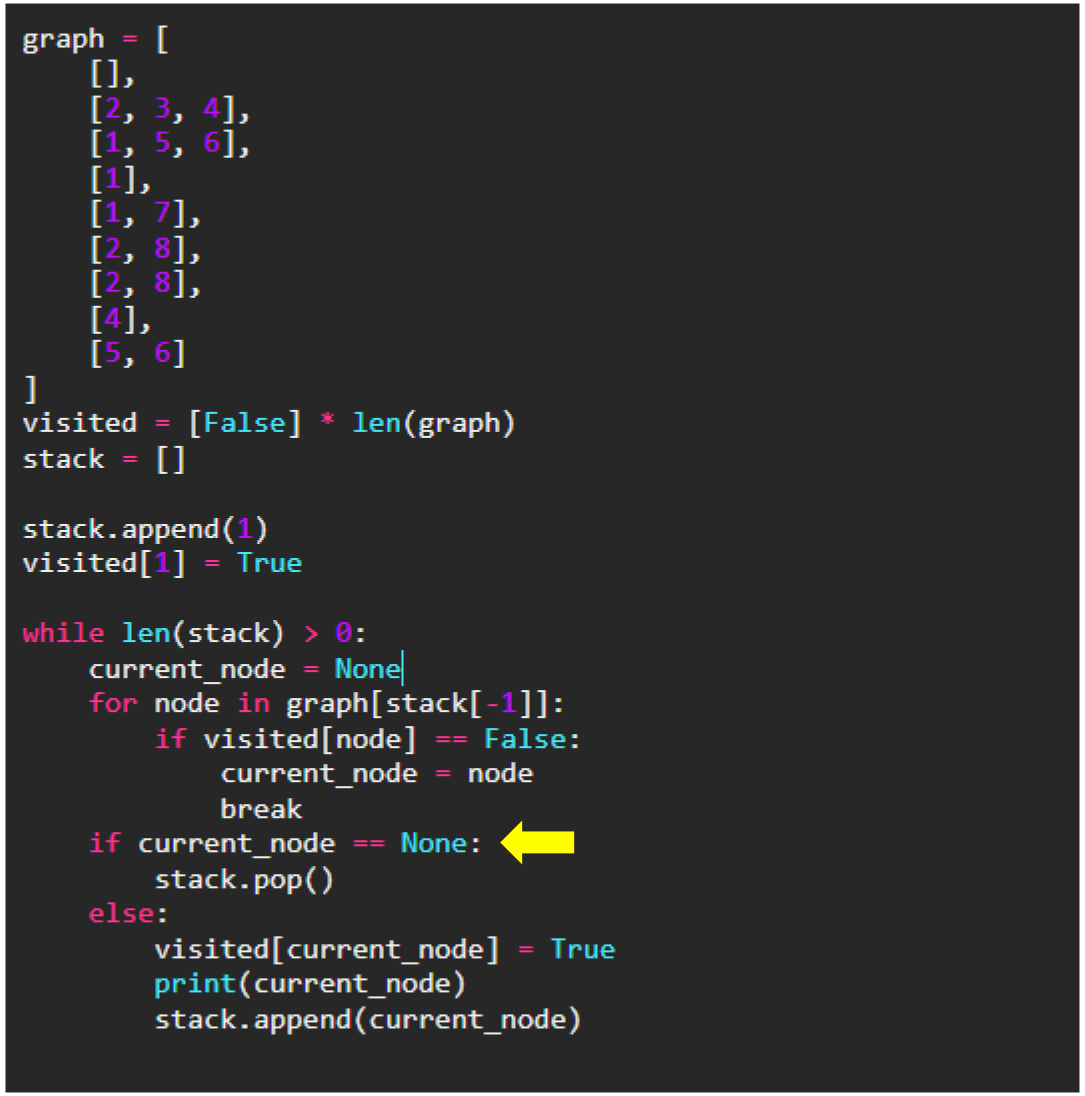




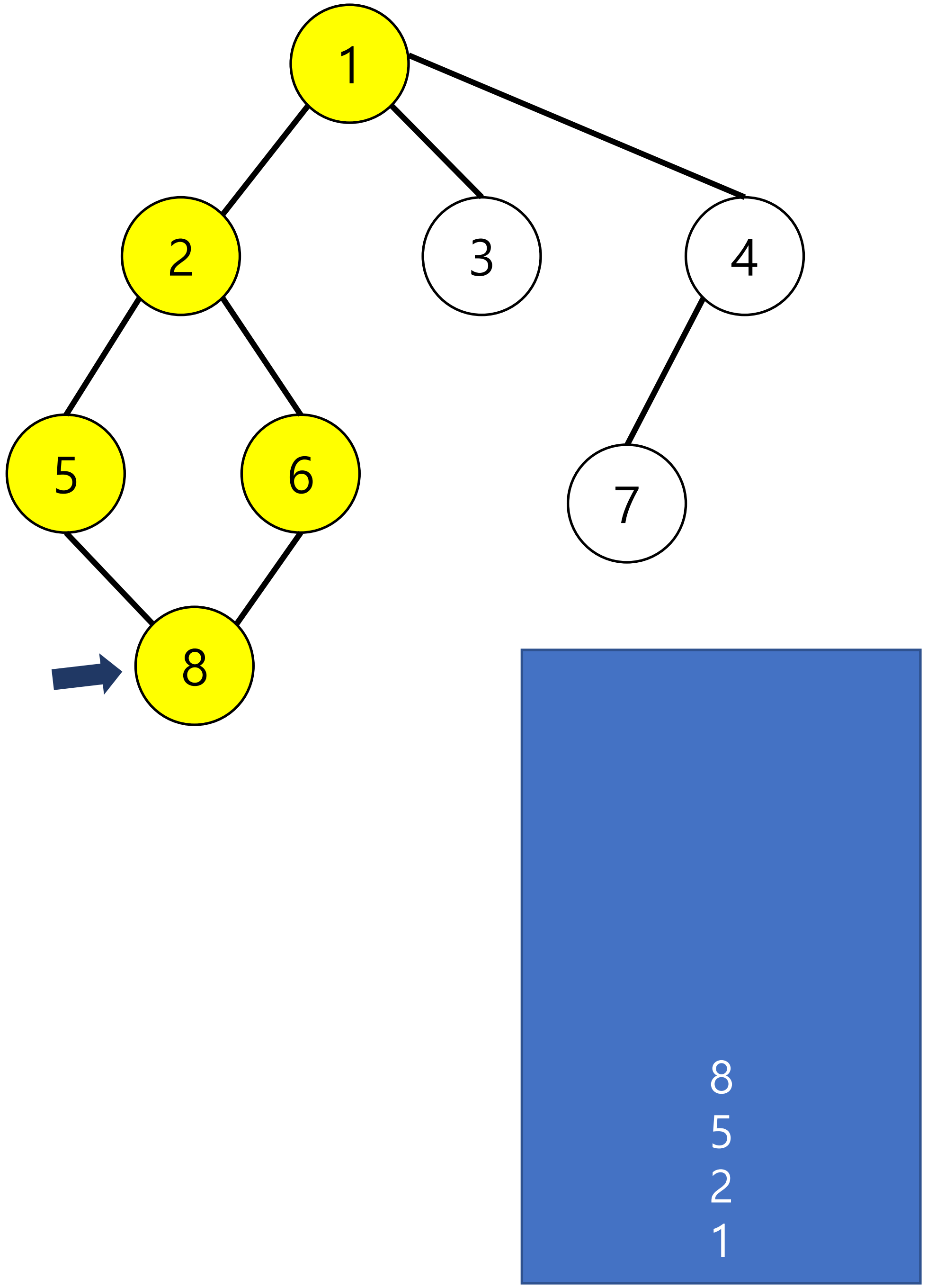


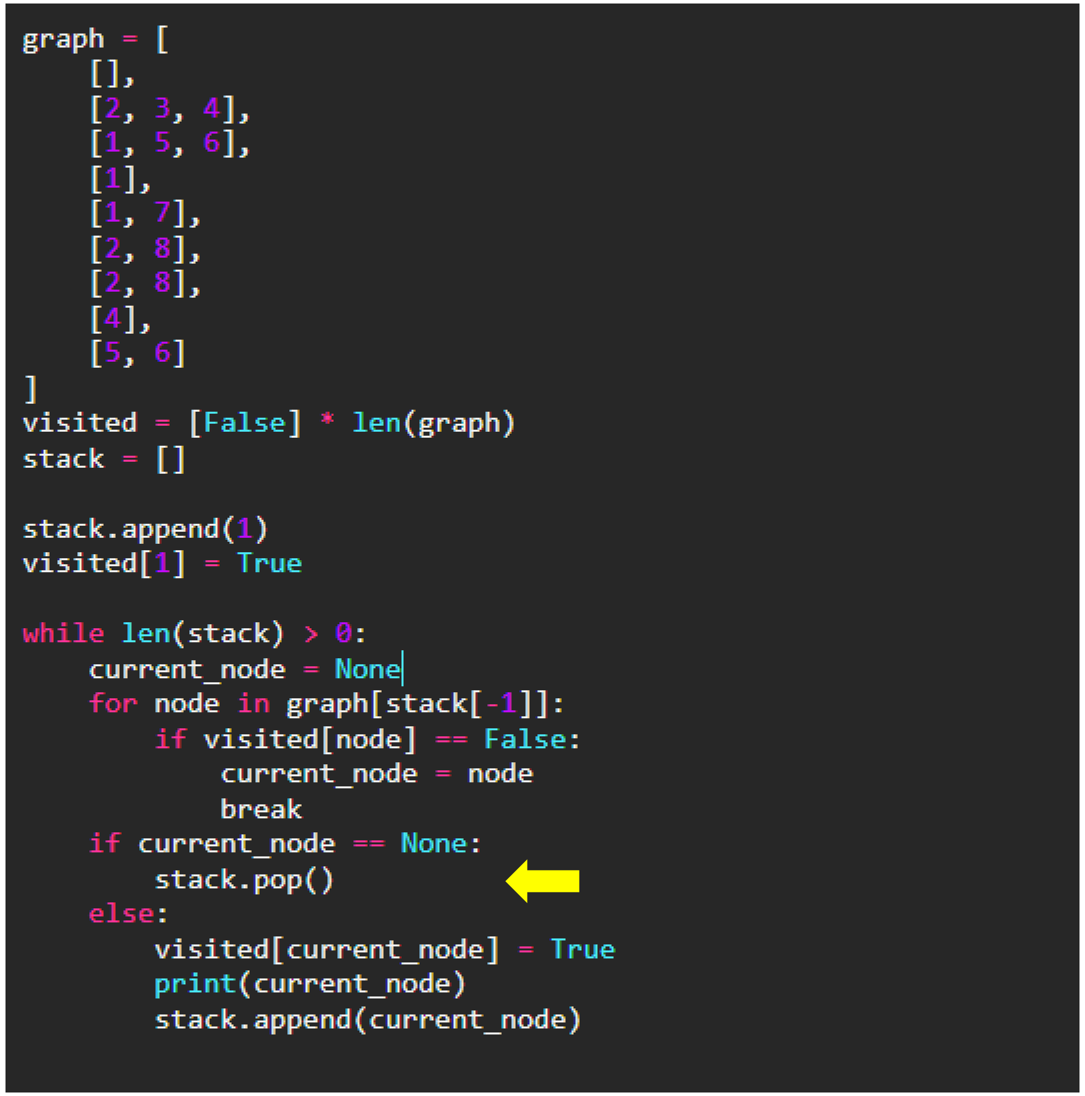
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]
visited = [False] * len(graph)
stack = []

stack.append(1)
visited[1] = True

while len(stack) > 0:
    current_node = None
    for node in graph[stack[-1]]:
        if visited[node] == False:
            current_node = node
            break
    if current_node == None:
        stack.pop()
    else:
        visited[current_node] = True
        print(current_node)
        stack.append(current_node)
```



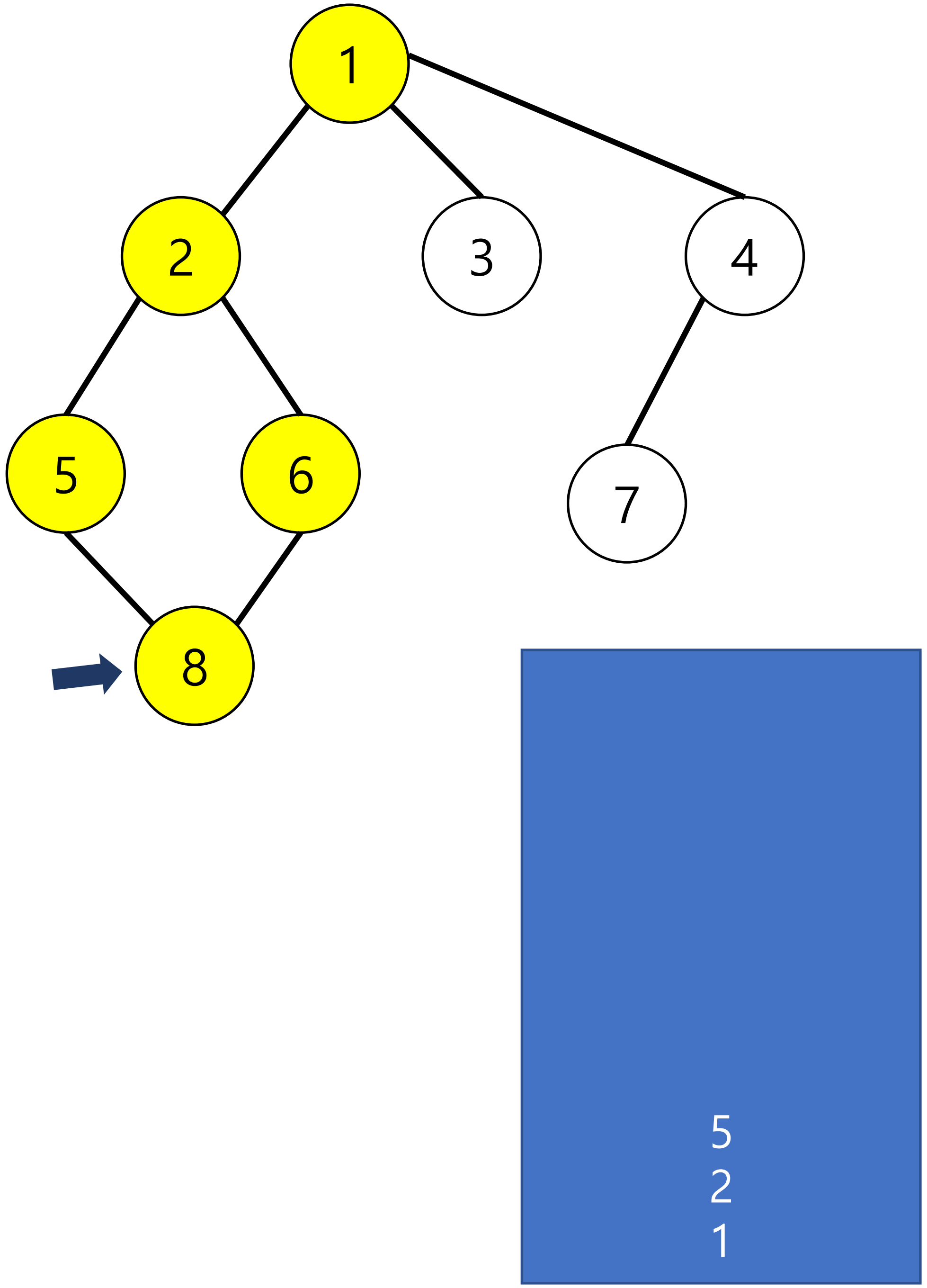


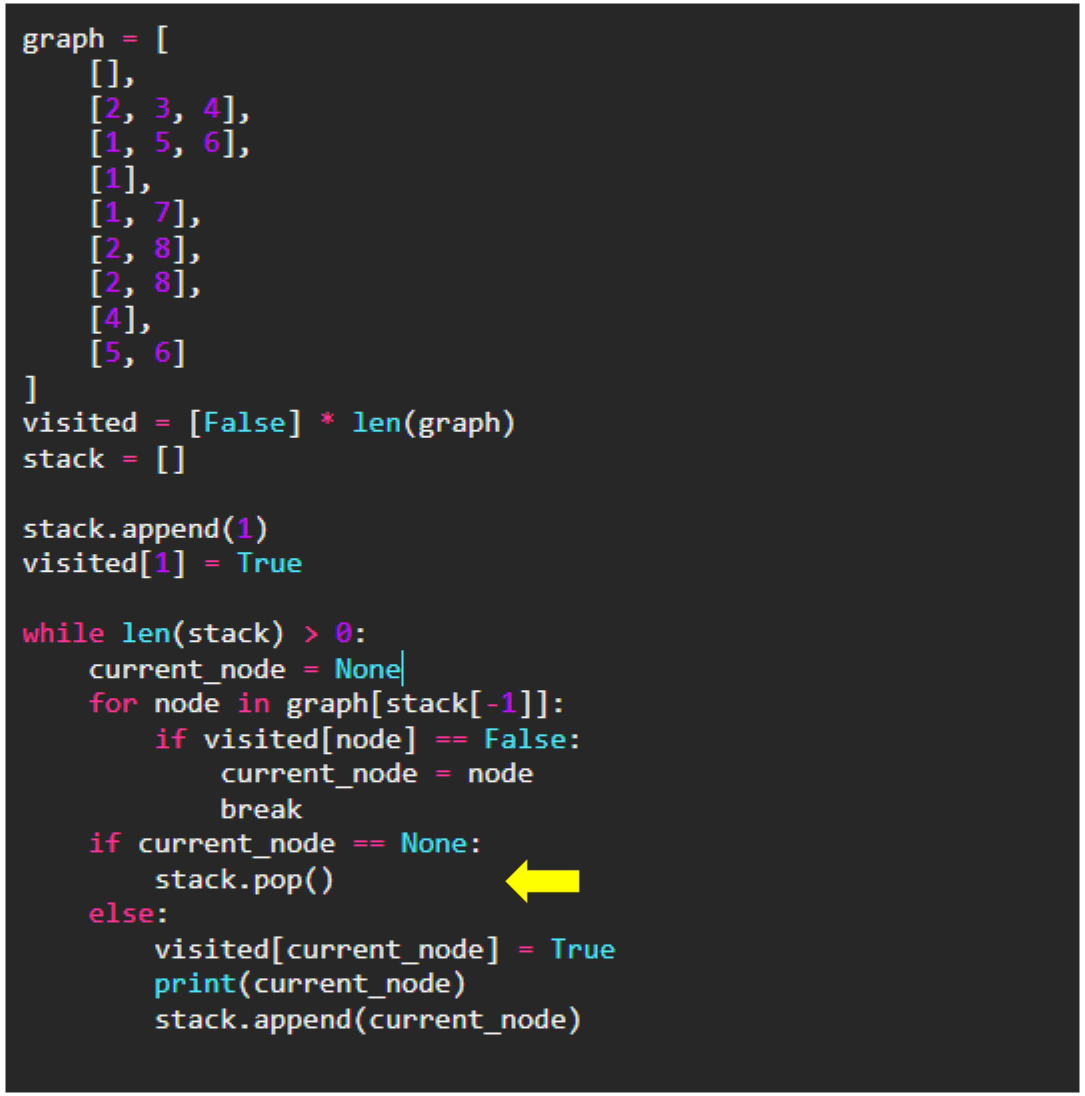
```python
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]
visited = [False] * len(graph)
stack = []

stack.append(1)
visited[1] = True

while len(stack) > 0:
    current_node = None
    for node in graph[stack[-1]]:
        if visited[node] == False:
            current_node = node
            break
    if current_node == None:
        stack.pop()
    else:
        visited[current_node] = True
        print(current_node)
        stack.append(current_node)
```



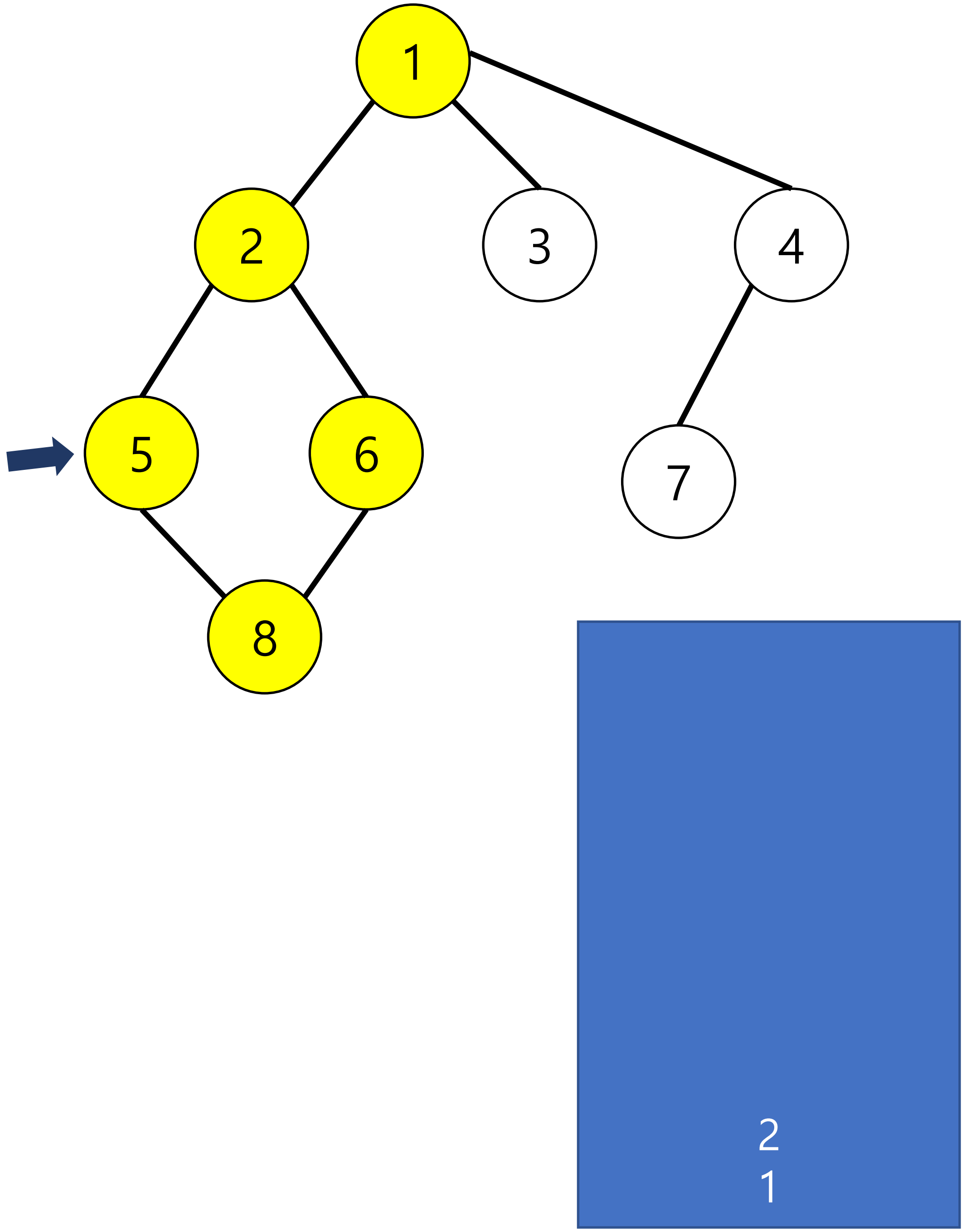


```python
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]
visited = [False] * len(graph)
stack = []

stack.append(1)
visited[1] = True

while len(stack) > 0:
    current_node = None
    for node in graph[stack[-1]]:
        if visited[node] == False:
            current_node = node
            break
    if current_node == None:
        stack.pop()
    else:
        visited[current_node] = True
        print(current_node)
        stack.append(current_node)
```

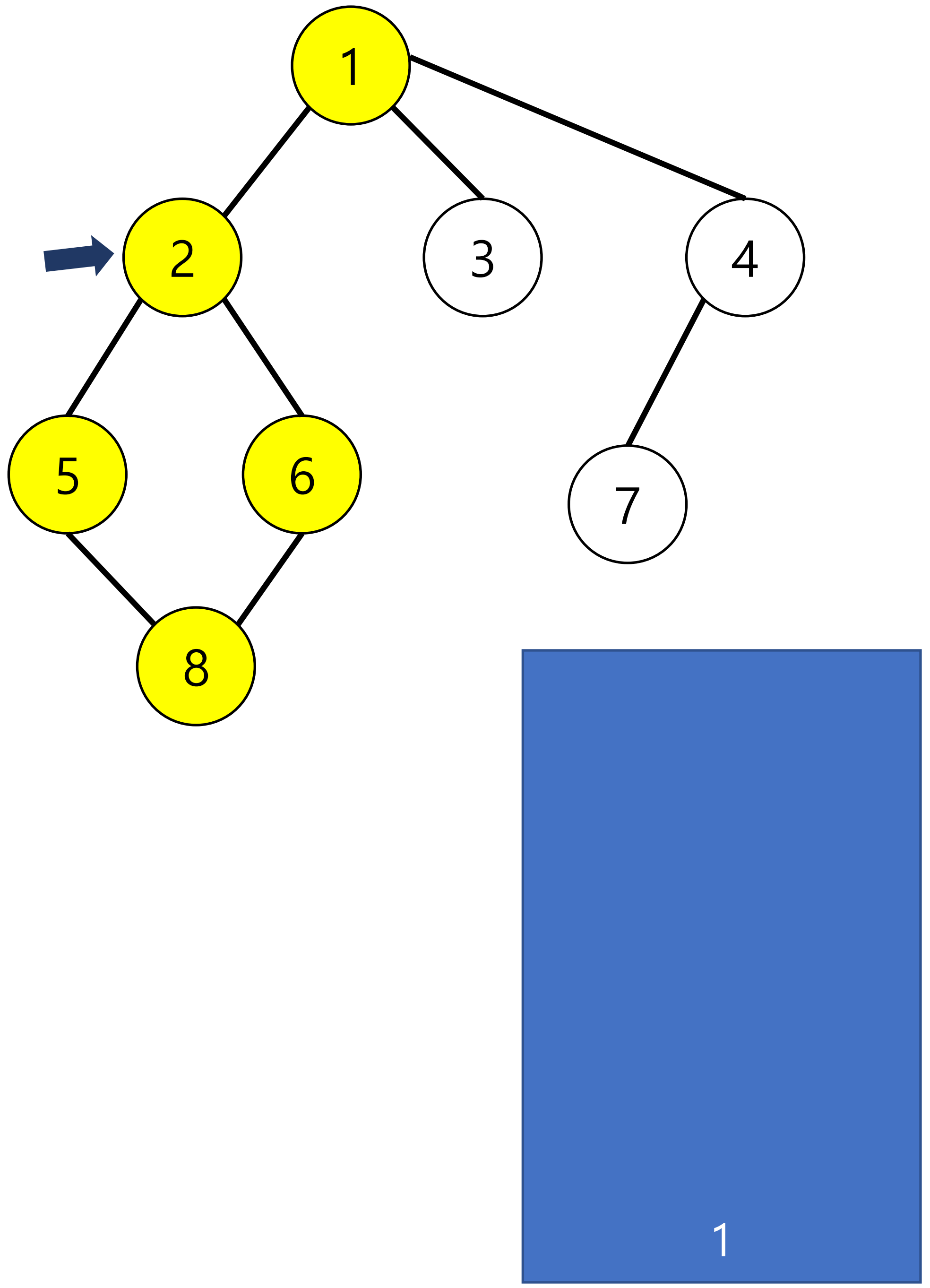


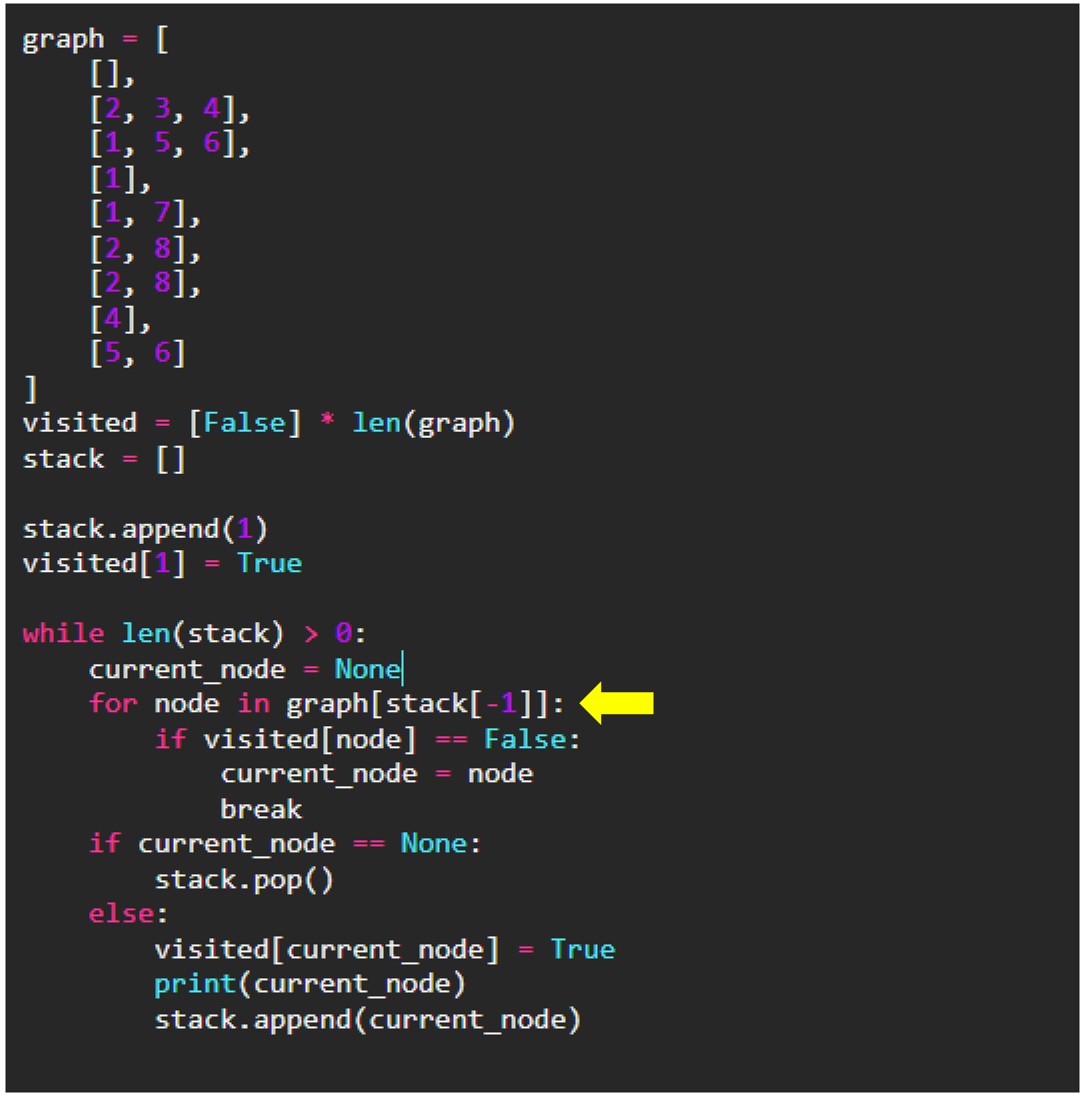


```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]
visited = [False] * len(graph)
stack = []

stack.append(1)
visited[1] = True

while len(stack) > 0:
    current_node = None
    for node in graph[stack[-1]]:
        if visited[node] == False:
            current_node = node
            break
    if current_node == None:
        stack.pop()
    else:
        visited[current_node] = True
        print(current_node)
        stack.append(current_node)
```

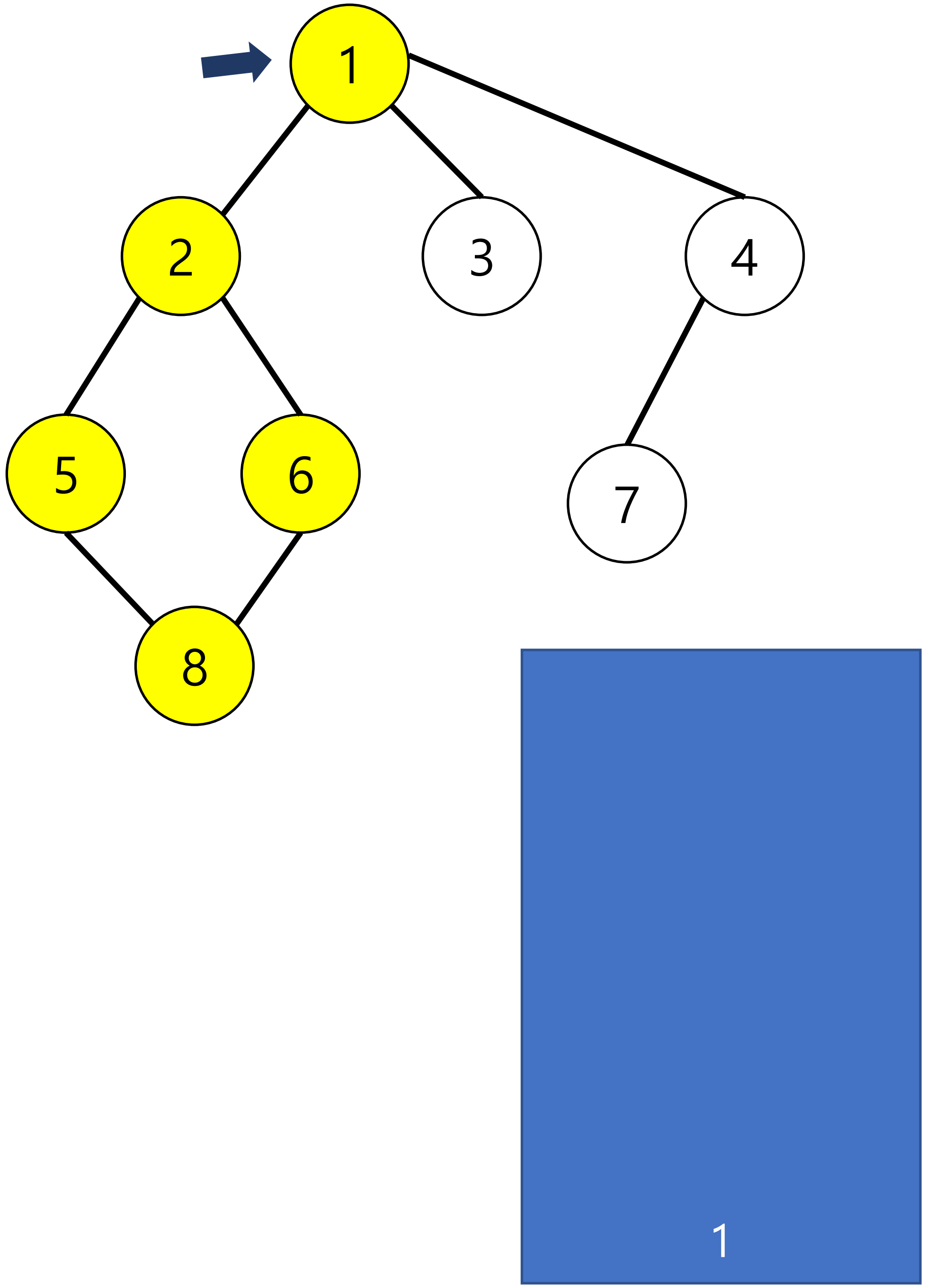


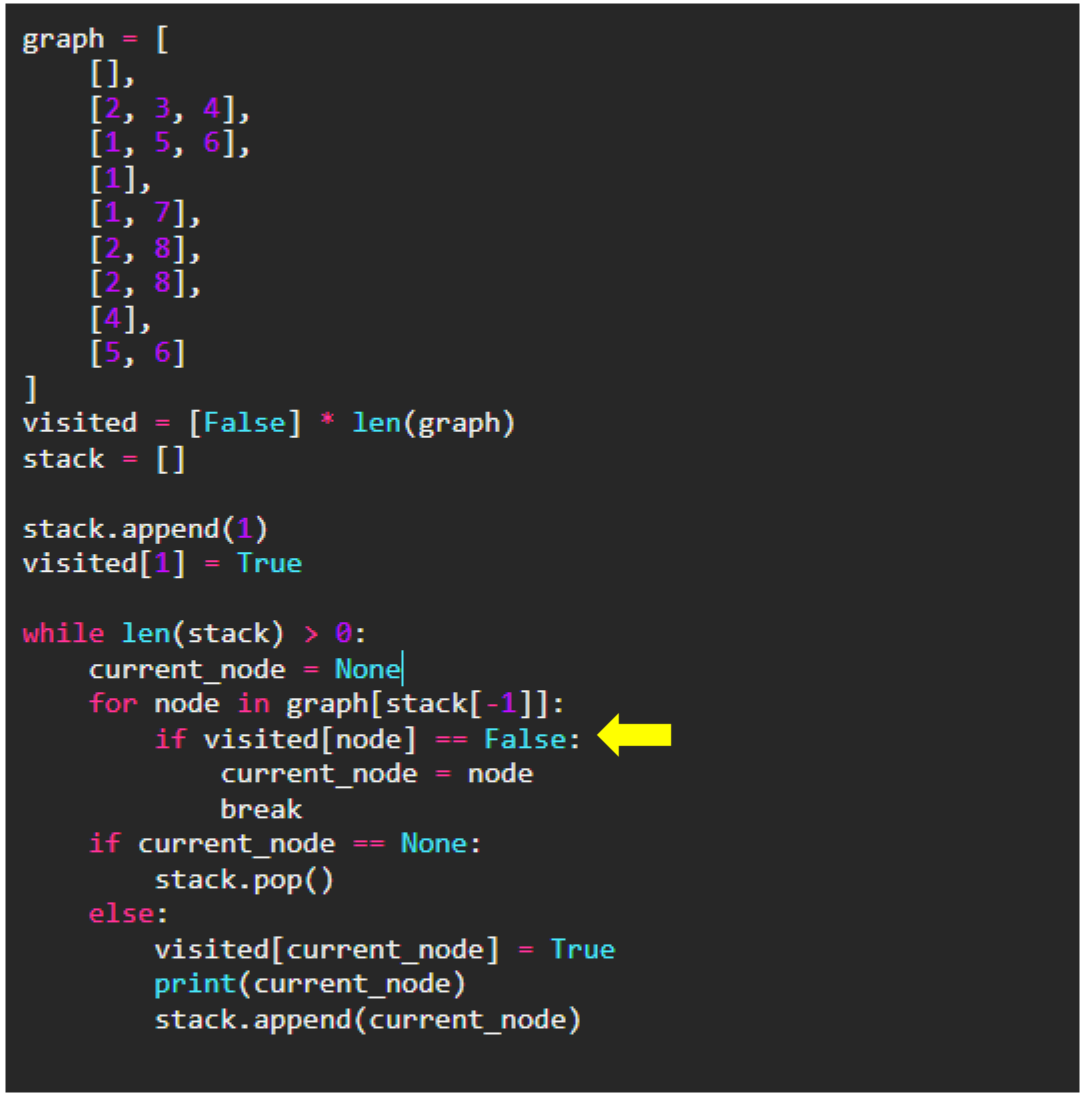
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]
visited = [False] * len(graph)
stack = []

stack.append(1)
visited[1] = True

while len(stack) > 0:
    current_node = None
    for node in graph[stack[-1]]:
        if visited[node] == False:
            current_node = node
            break
    if current_node == None:
        stack.pop()
    else:
        visited[current_node] = True
        print(current_node)
        stack.append(current_node)
```



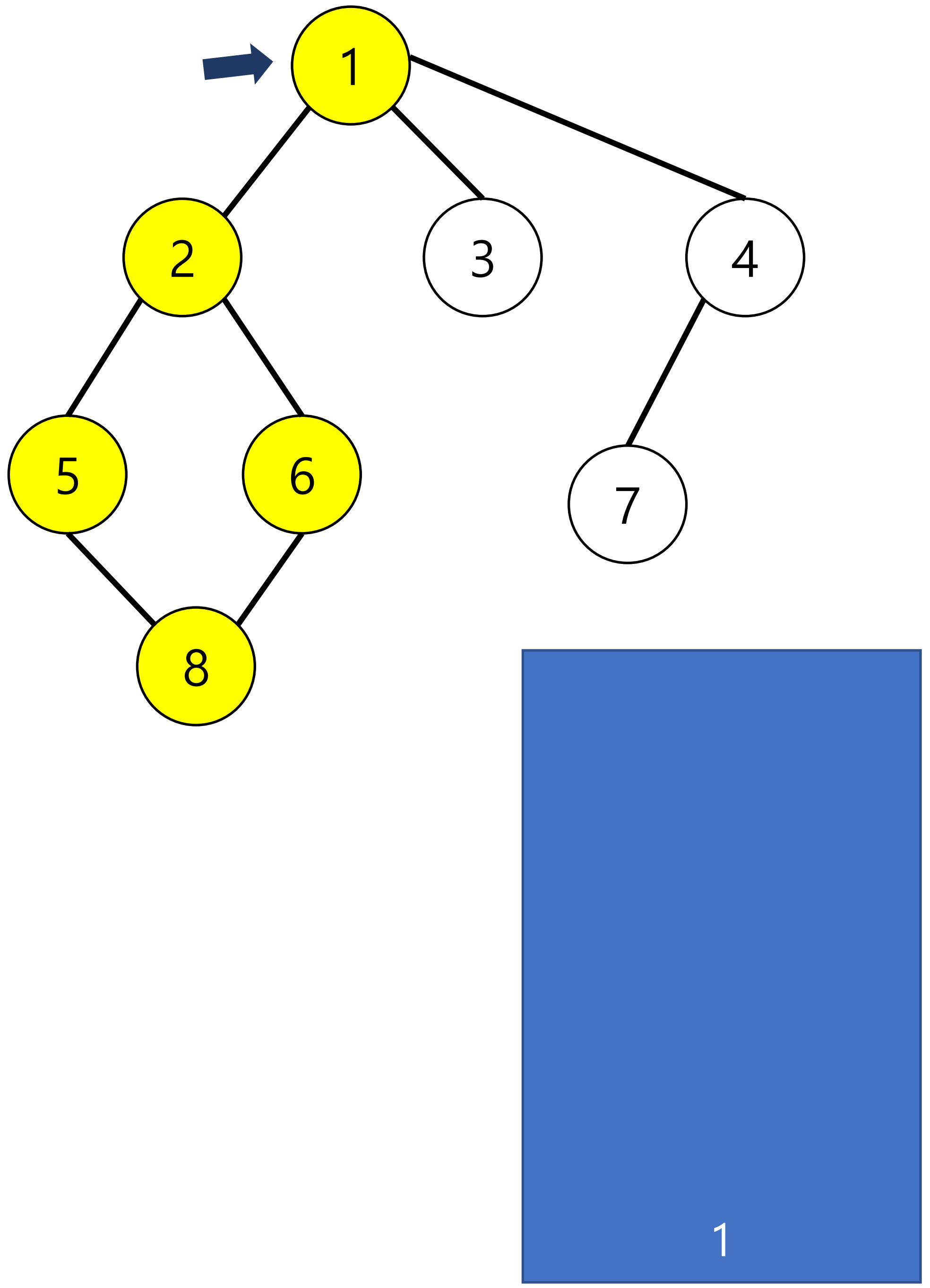


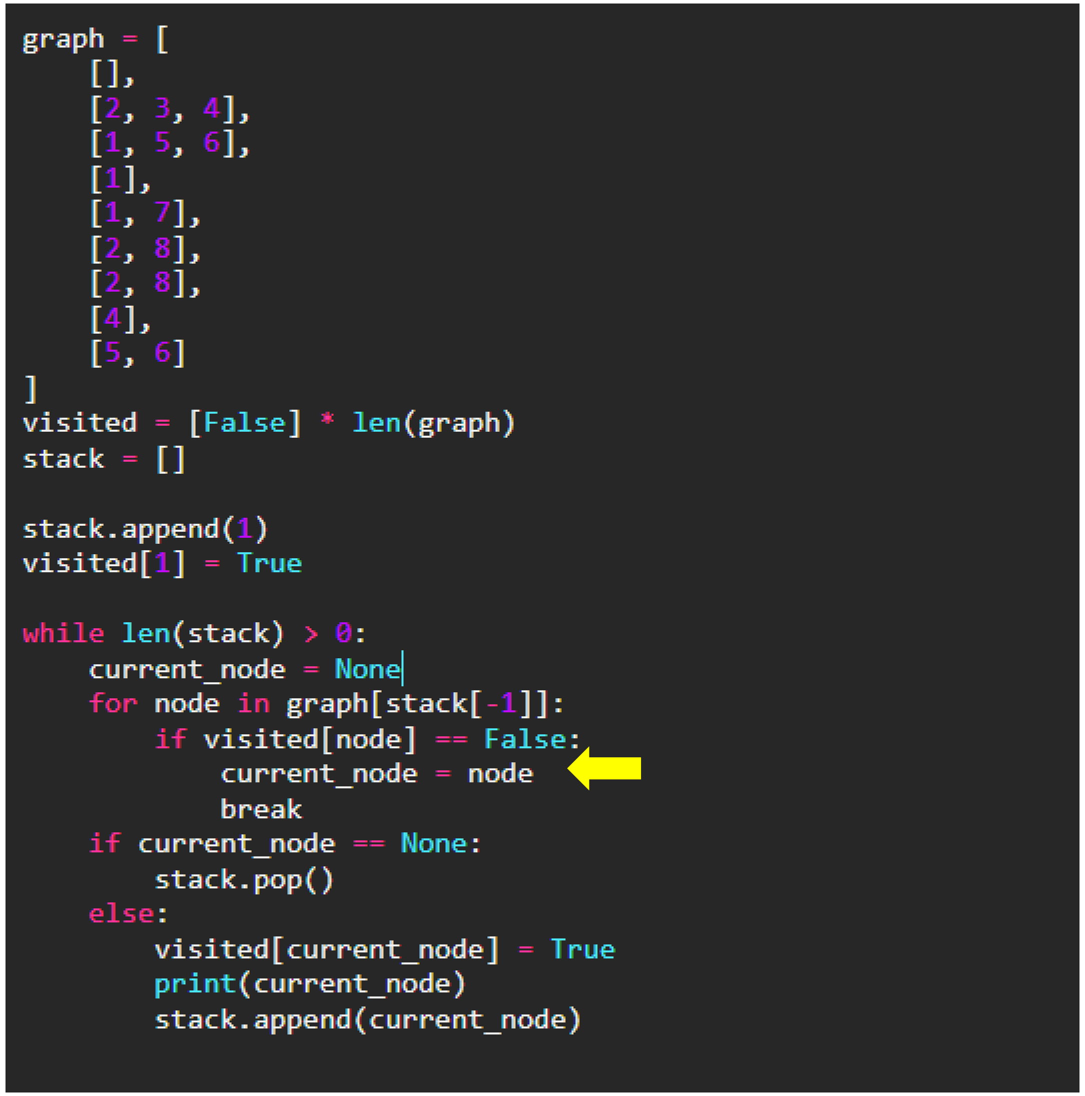
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]
visited = [False] * len(graph)
stack = []

stack.append(1)
visited[1] = True

while len(stack) > 0:
    current_node = None
    for node in graph[stack[-1]]:
        if visited[node] == False:
            current_node = node
            break
    if current_node == None:
        stack.pop()
    else:
        visited[current_node] = True
        print(current_node)
        stack.append(current_node)
```



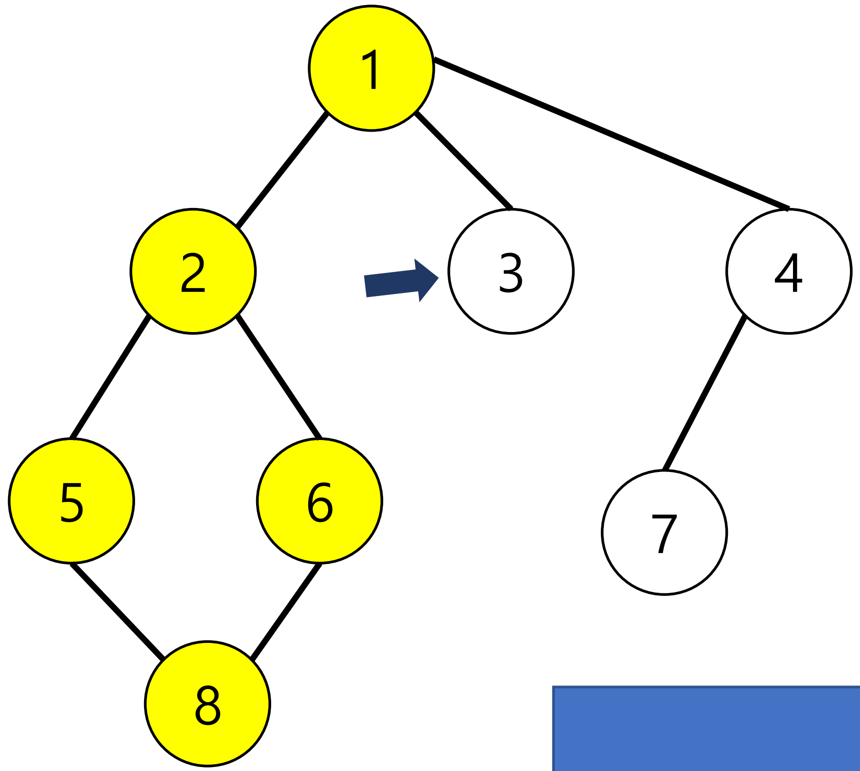


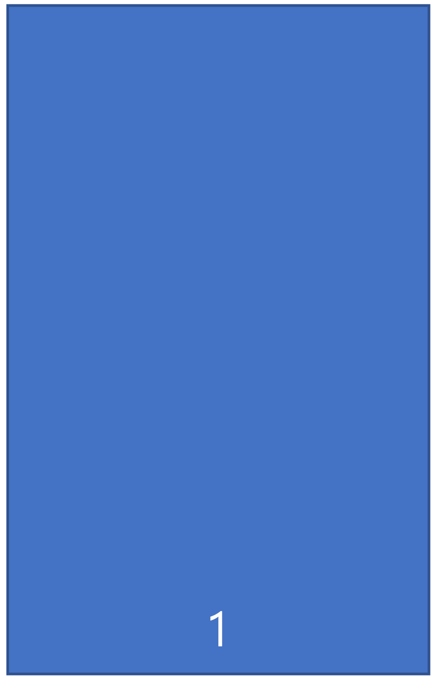
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]
visited = [False] * len(graph)
stack = []

stack.append(1)
visited[1] = True

while len(stack) > 0:
    current_node = None
    for node in graph[stack[-1]]:
        if visited[node] == False:
            current_node = node
            break
    if current_node == None:
        stack.pop()
    else:
        visited[current_node] = True
        print(current_node)
        stack.append(current_node)
```

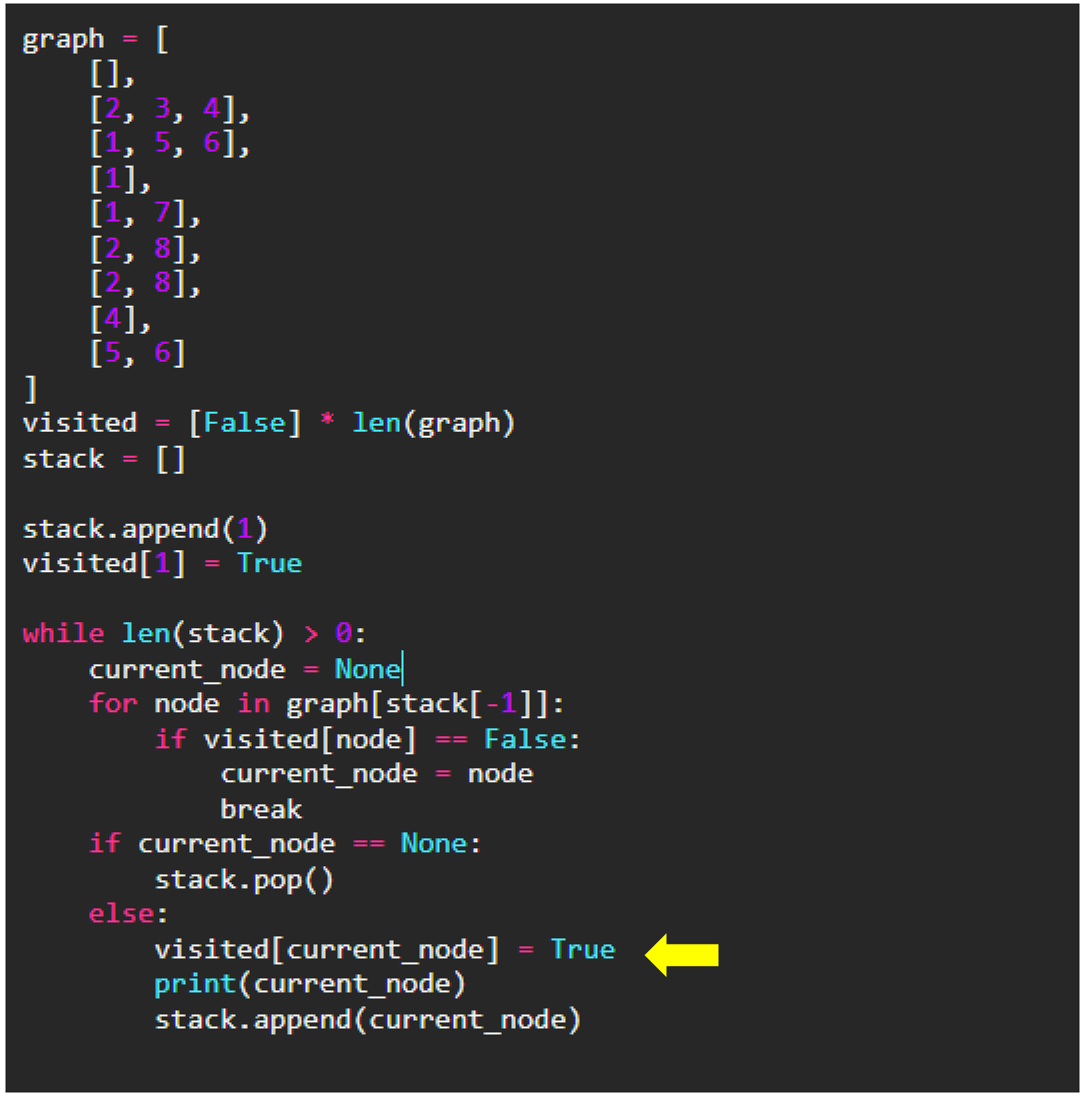




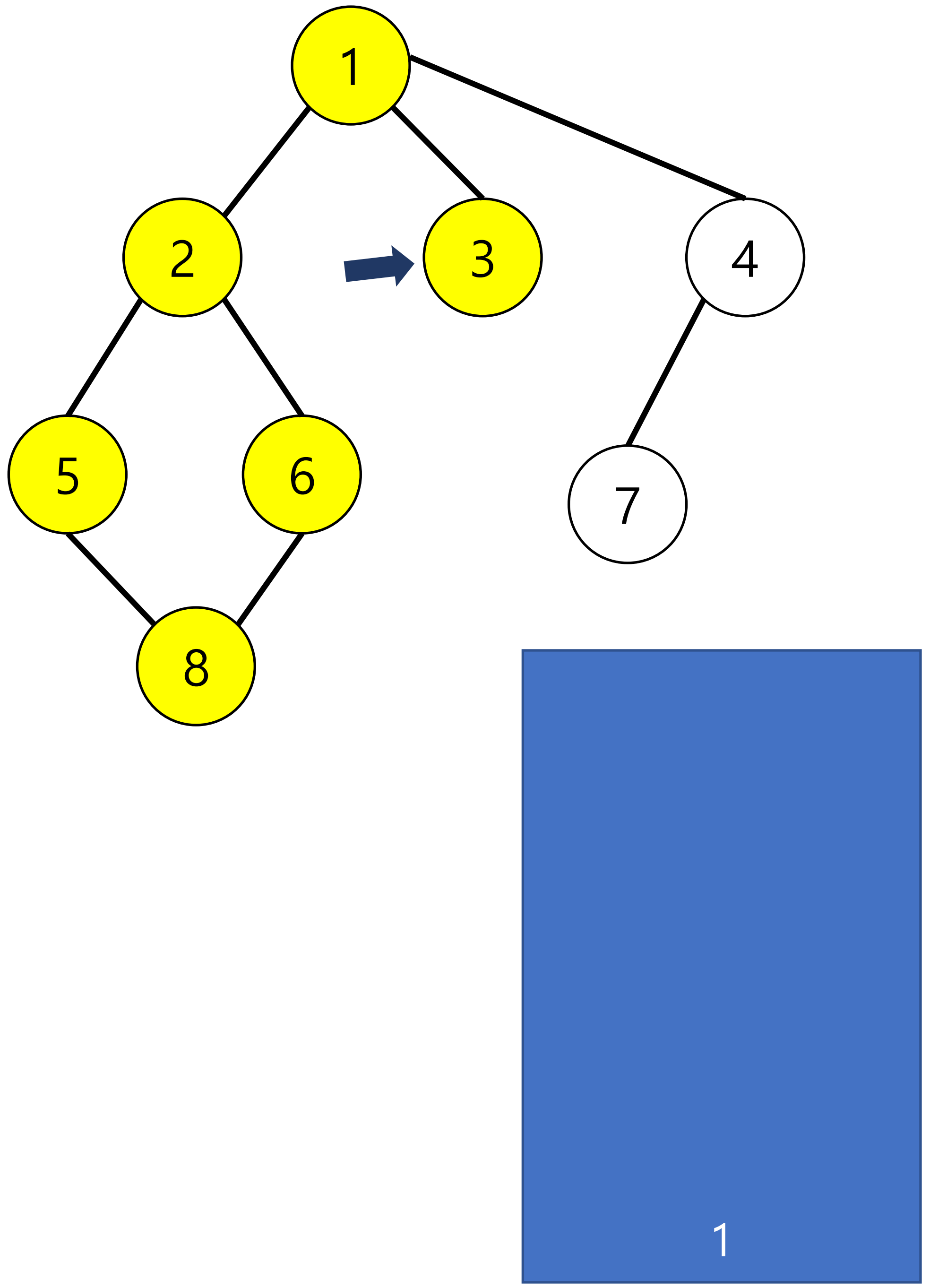
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]
visited = [False] * len(graph)
stack = []

stack.append(1)
visited[1] = True

while len(stack) > 0:
    current_node = None
    for node in graph[stack[-1]]:
        if visited[node] == False:
            current_node = node
            break
    if current_node == None:
        stack.pop()
    else:
        visited[current_node] = True
        print(current_node)
        stack.append(current_node)
```

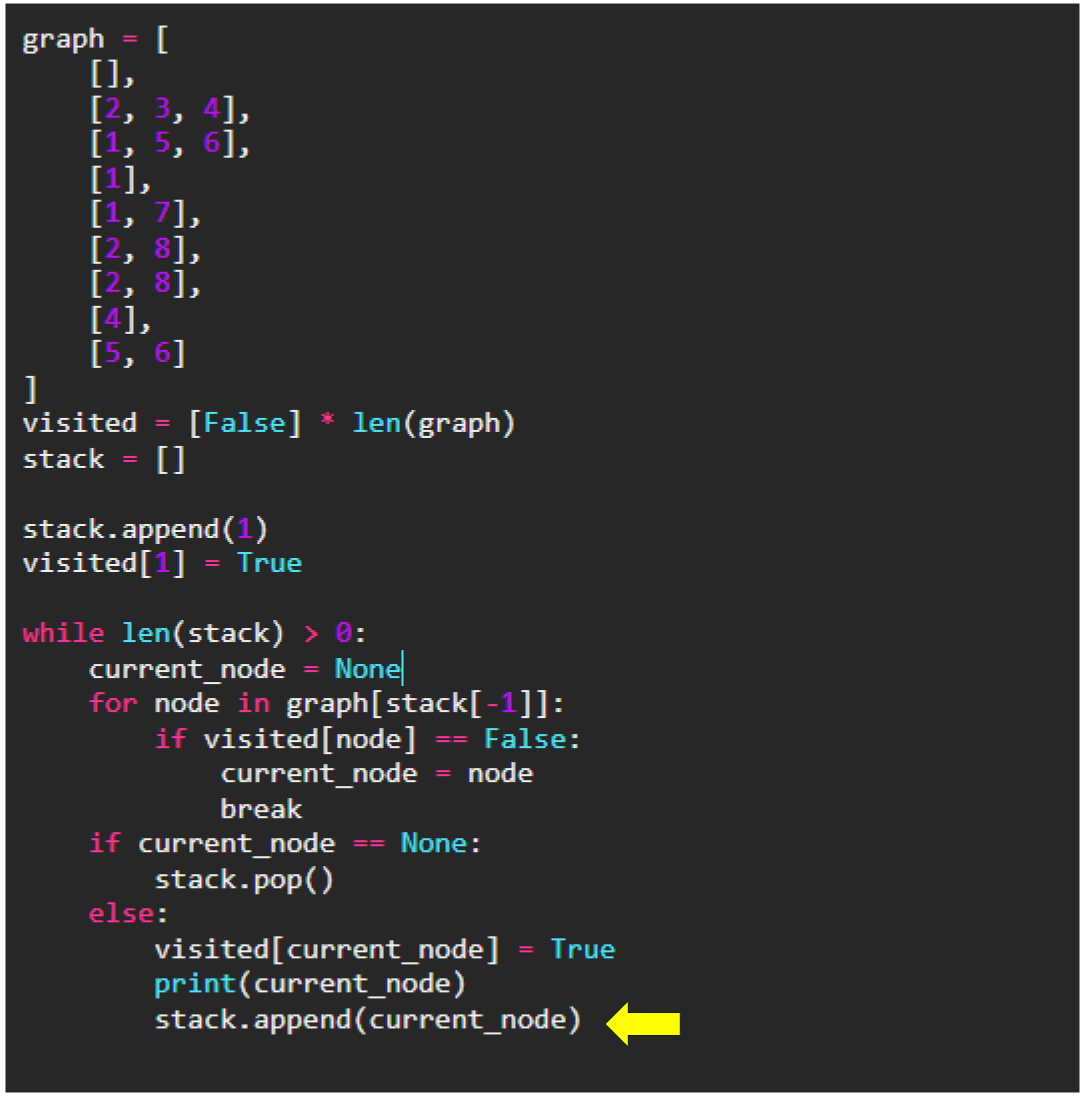




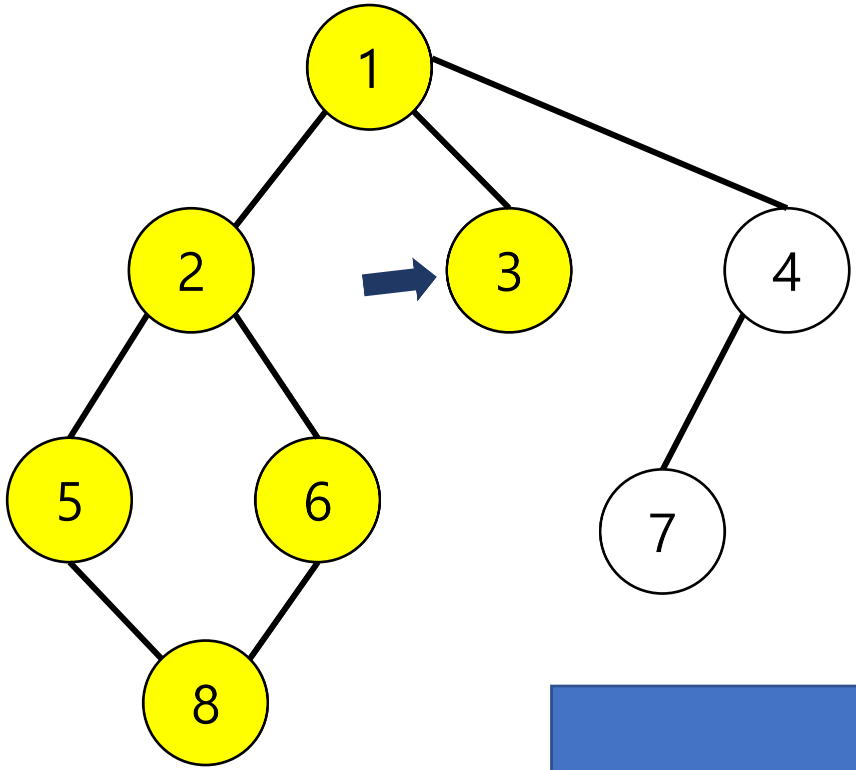


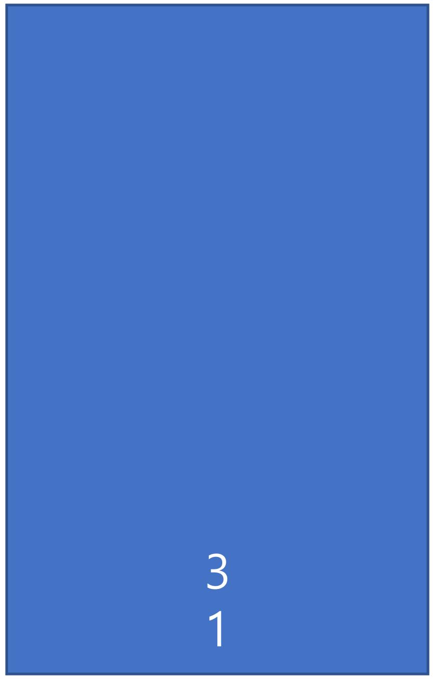
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]
visited = [False] * len(graph)
stack = []

stack.append(1)
visited[1] = True

while len(stack) > 0:
    current_node = None
    for node in graph[stack[-1]]:
        if visited[node] == False:
            current_node = node
            break
    if current_node == None:
        stack.pop()
    else:
        visited[current_node] = True
        print(current_node)
        stack.append(current_node)
```



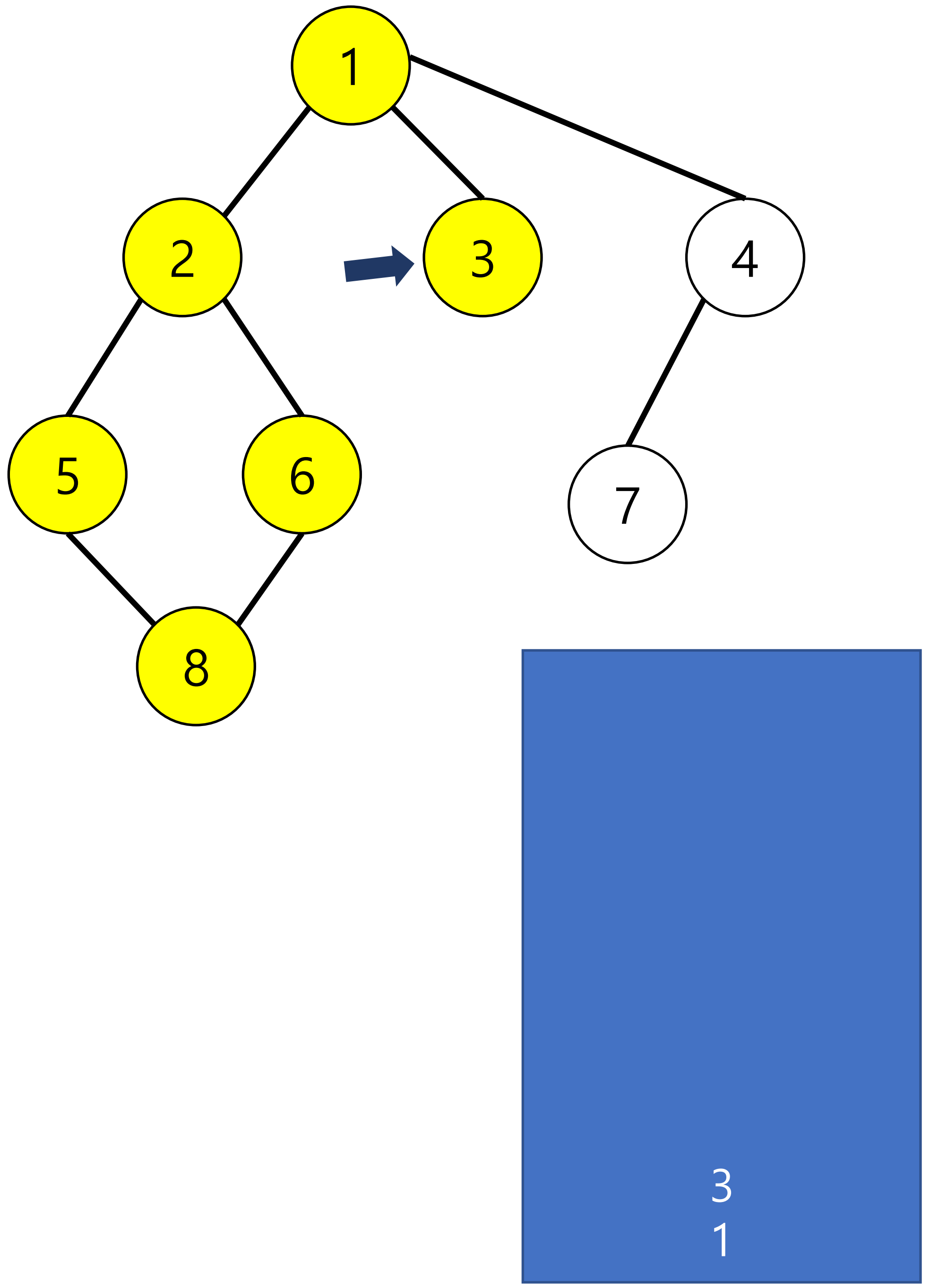


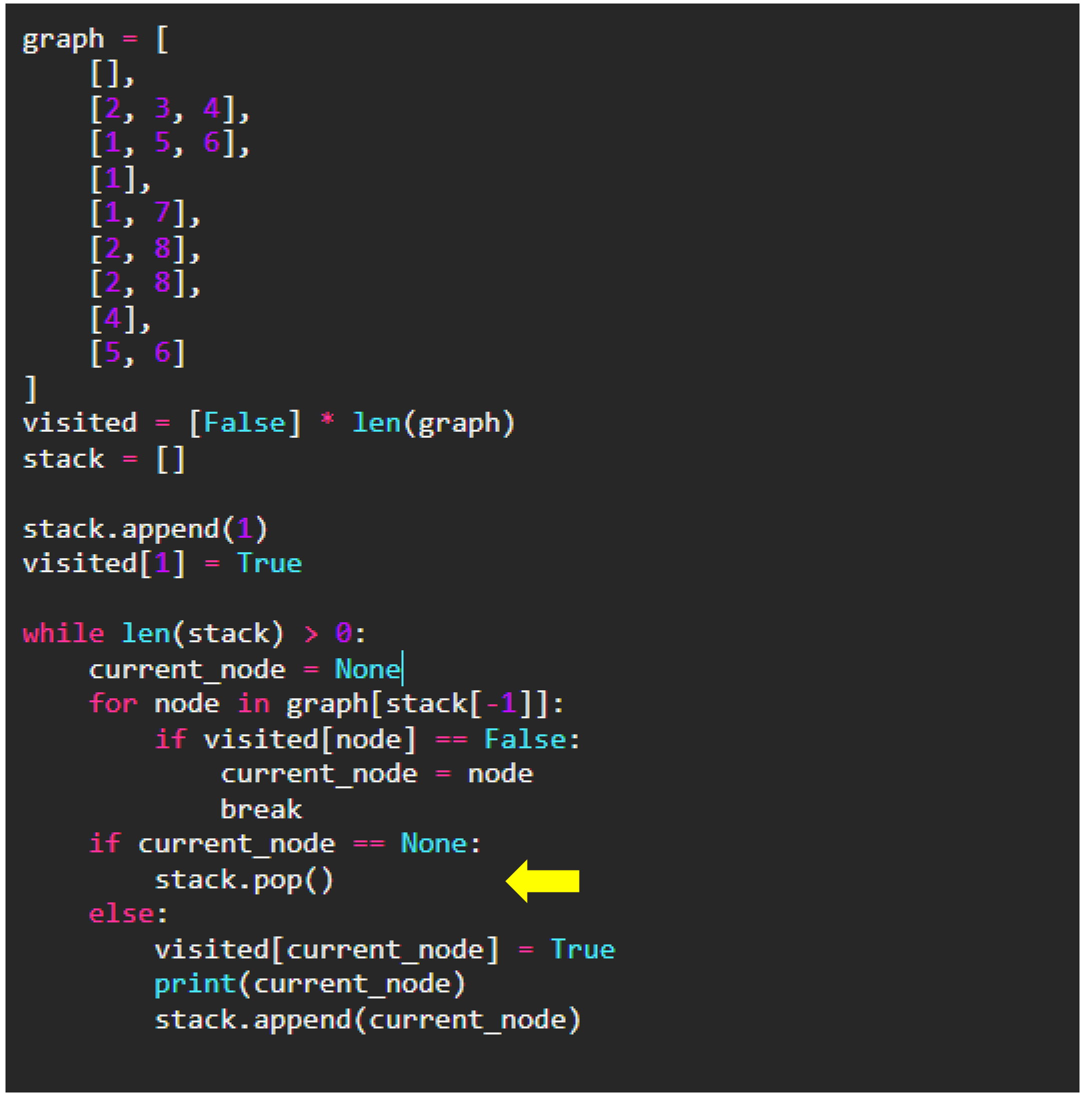
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]
visited = [False] * len(graph)
stack = []

stack.append(1)
visited[1] = True

while len(stack) > 0:
    current_node = None
    for node in graph[stack[-1]]:
        if visited[node] == False:
            current_node = node
            break
    if current_node == None:
        stack.pop()
    else:
        visited[current_node] = True
        print(current_node)
        stack.append(current_node)
```



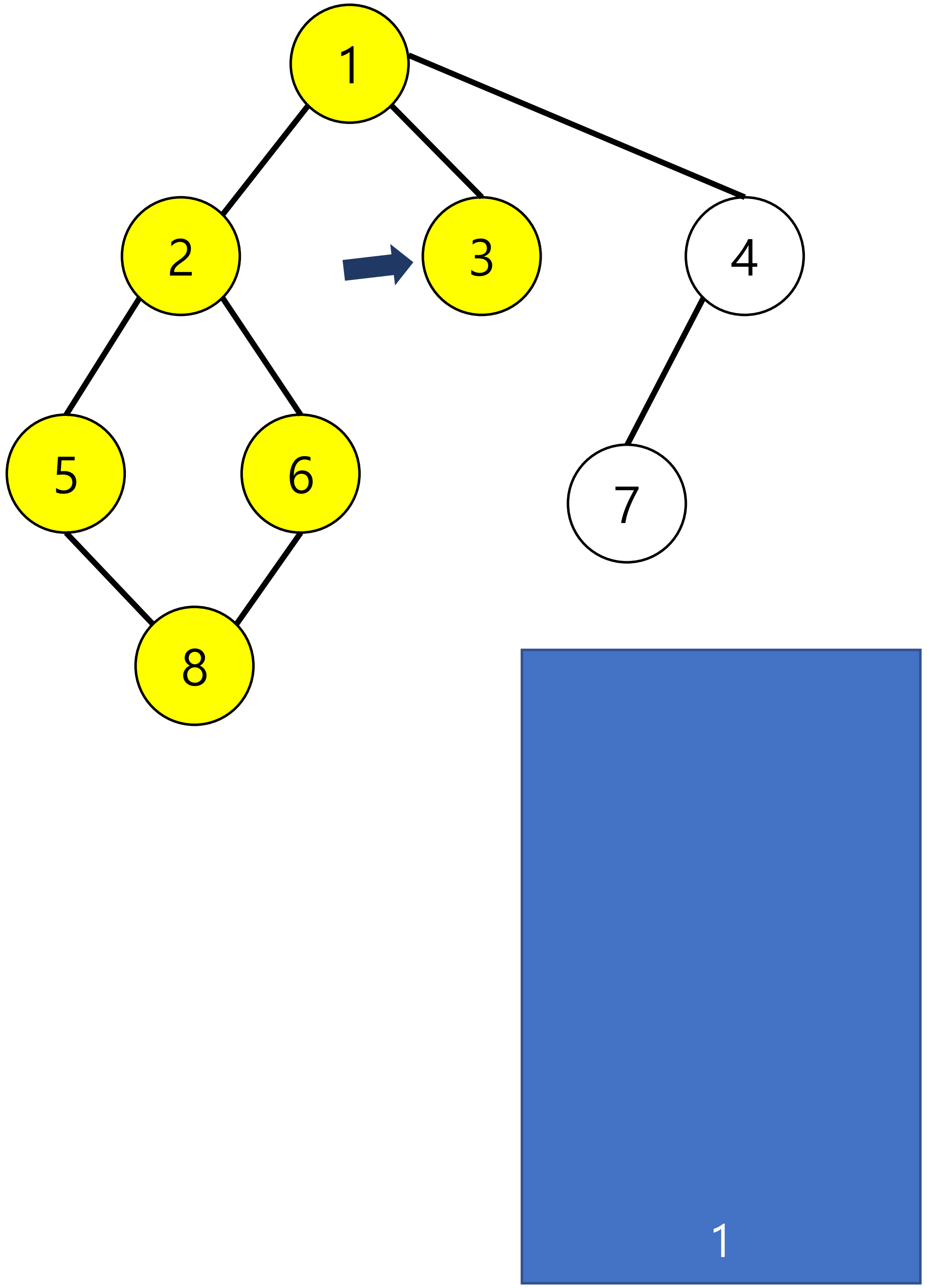


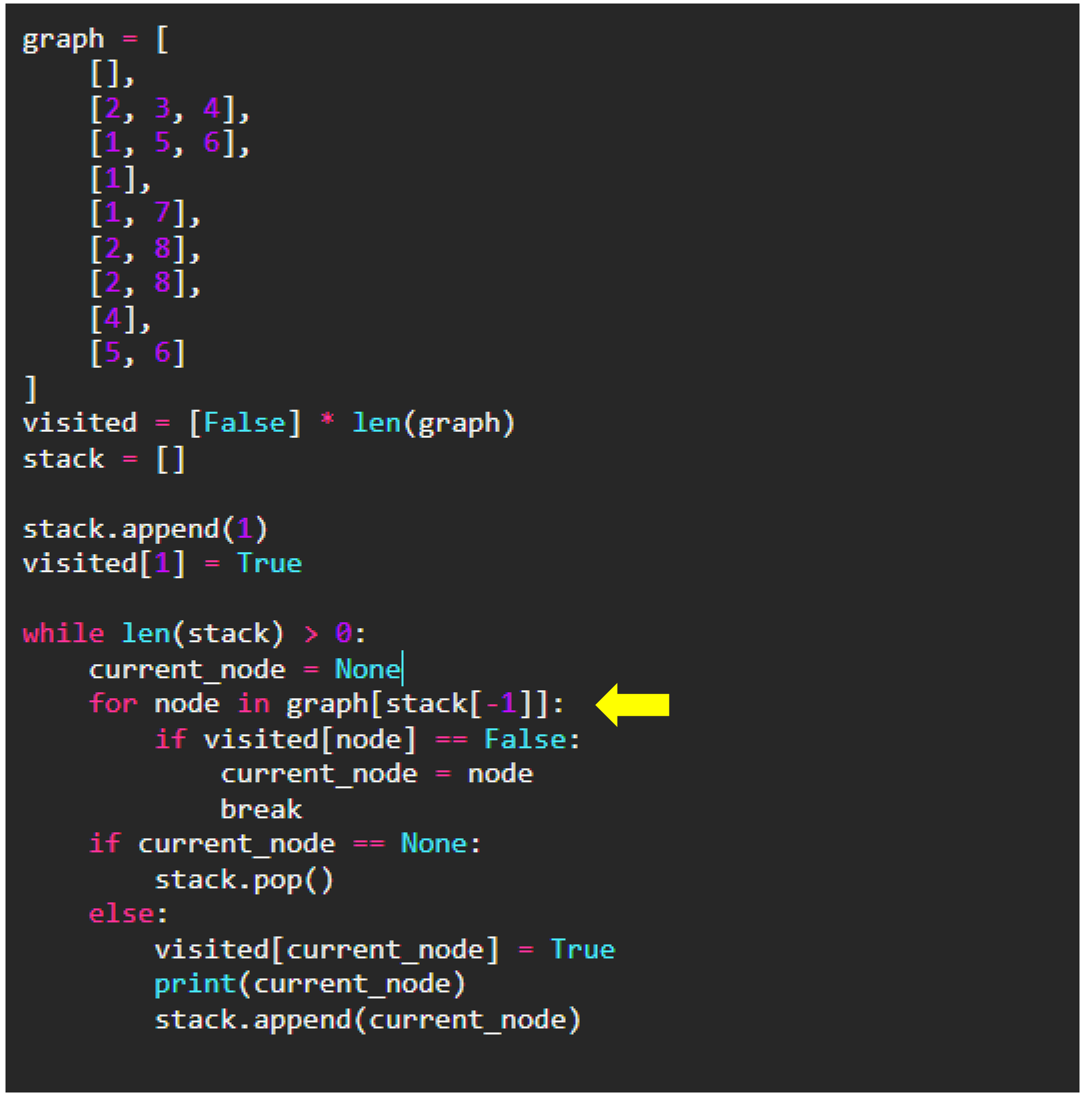
```python
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]
visited = [False] * len(graph)
stack = []

stack.append(1)
visited[1] = True

while len(stack) > 0:
    current_node = None
    for node in graph[stack[-1]]:
        if visited[node] == False:
            current_node = node
            break
    if current_node == None:
        stack.pop()
    else:
        visited[current_node] = True
        print(current_node)
        stack.append(current_node)
```



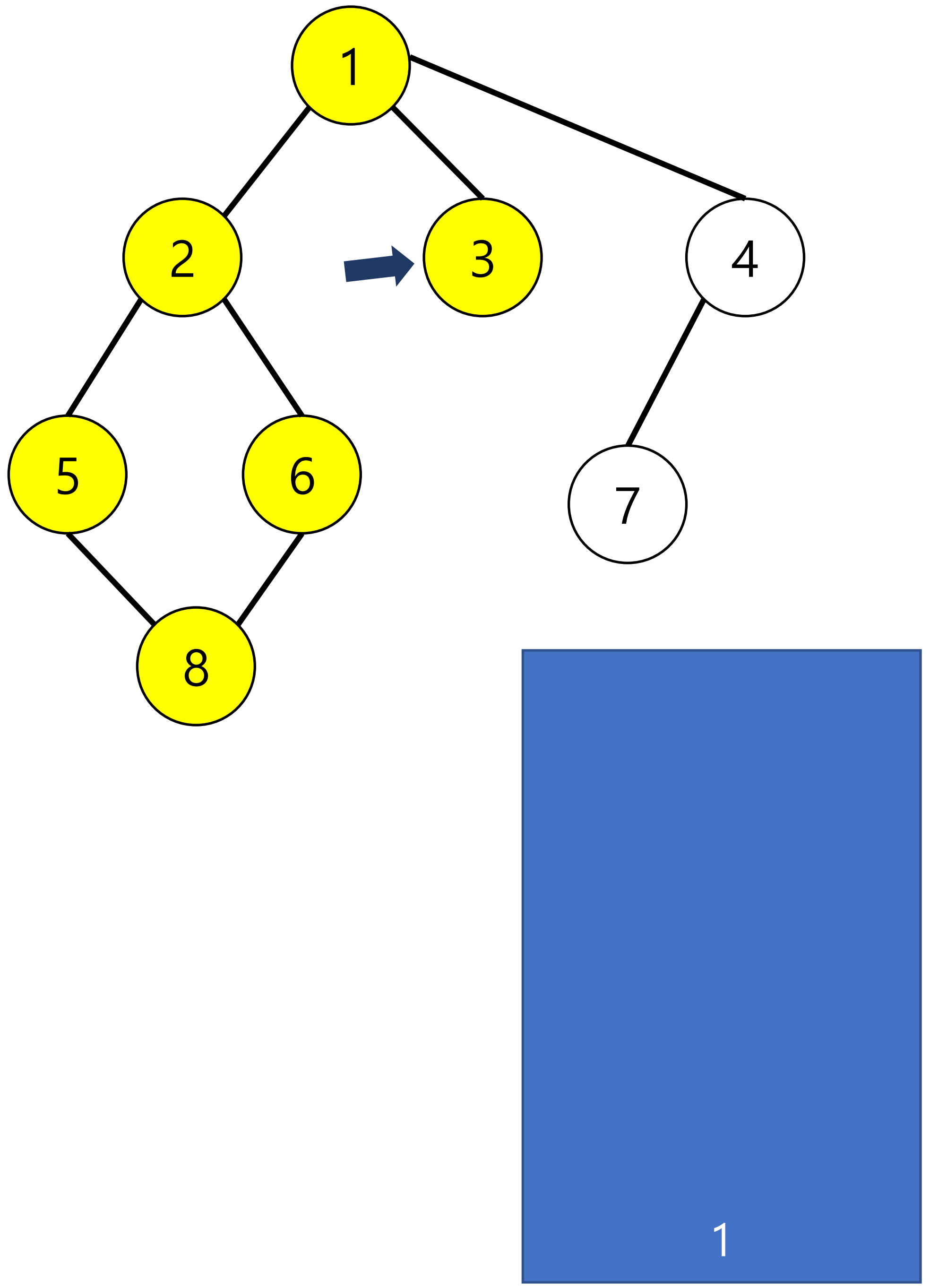


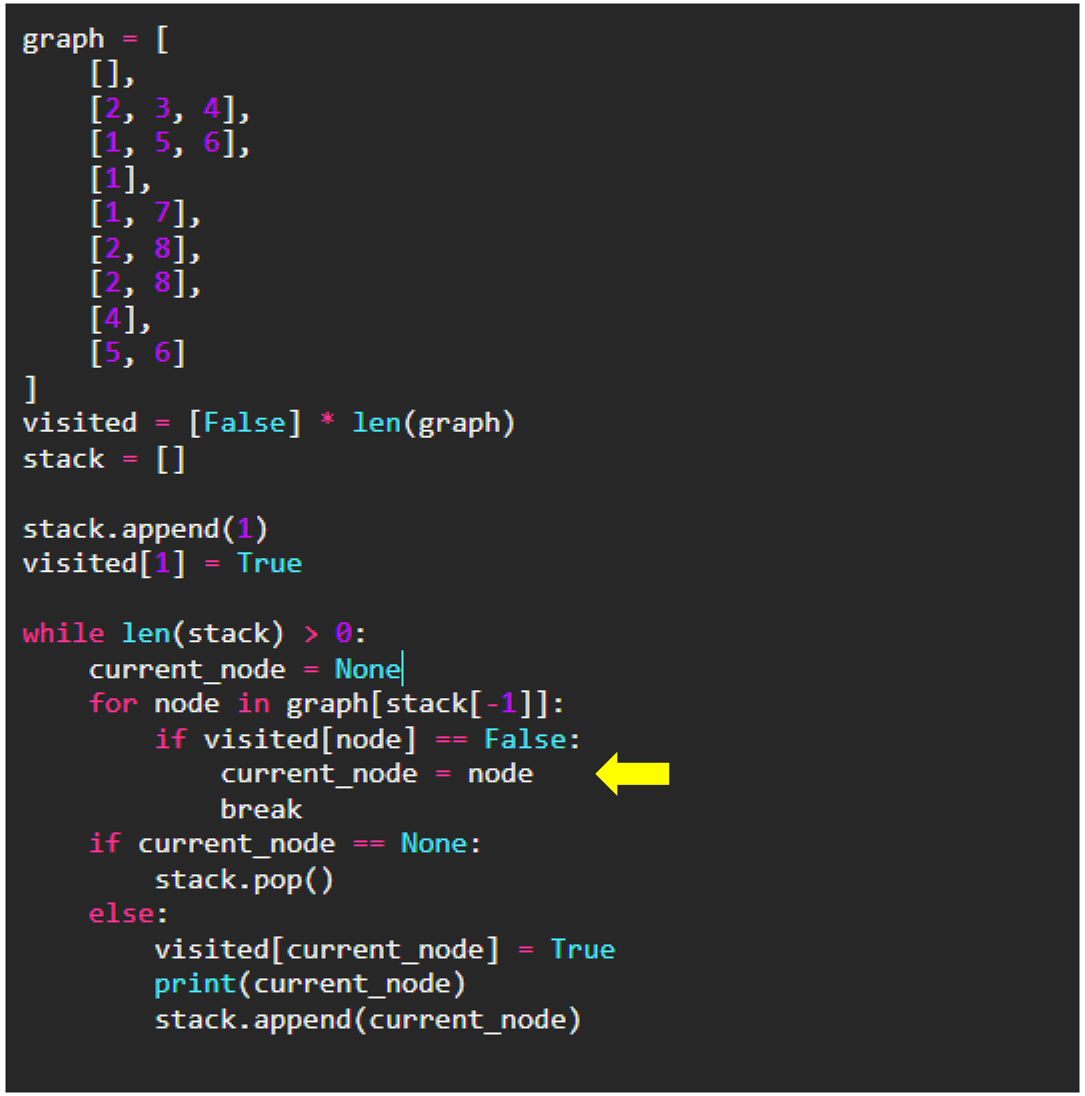
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]
visited = [False] * len(graph)
stack = []

stack.append(1)
visited[1] = True

while len(stack) > 0:
    current_node = None
    for node in graph[stack[-1]]:
        if visited[node] == False:
            current_node = node
            break
    if current_node == None:
        stack.pop()
    else:
        visited[current_node] = True
        print(current_node)
        stack.append(current_node)
```



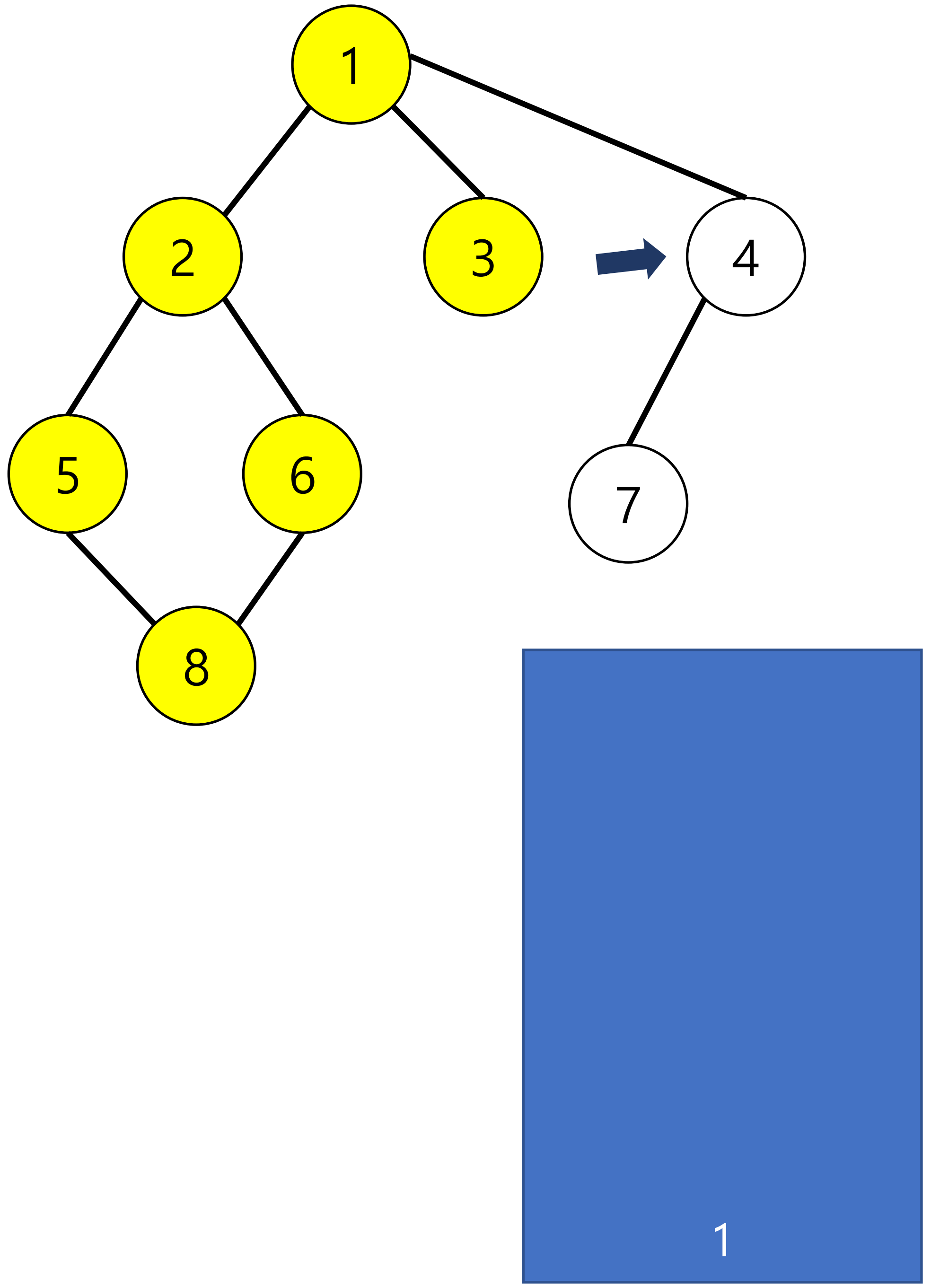


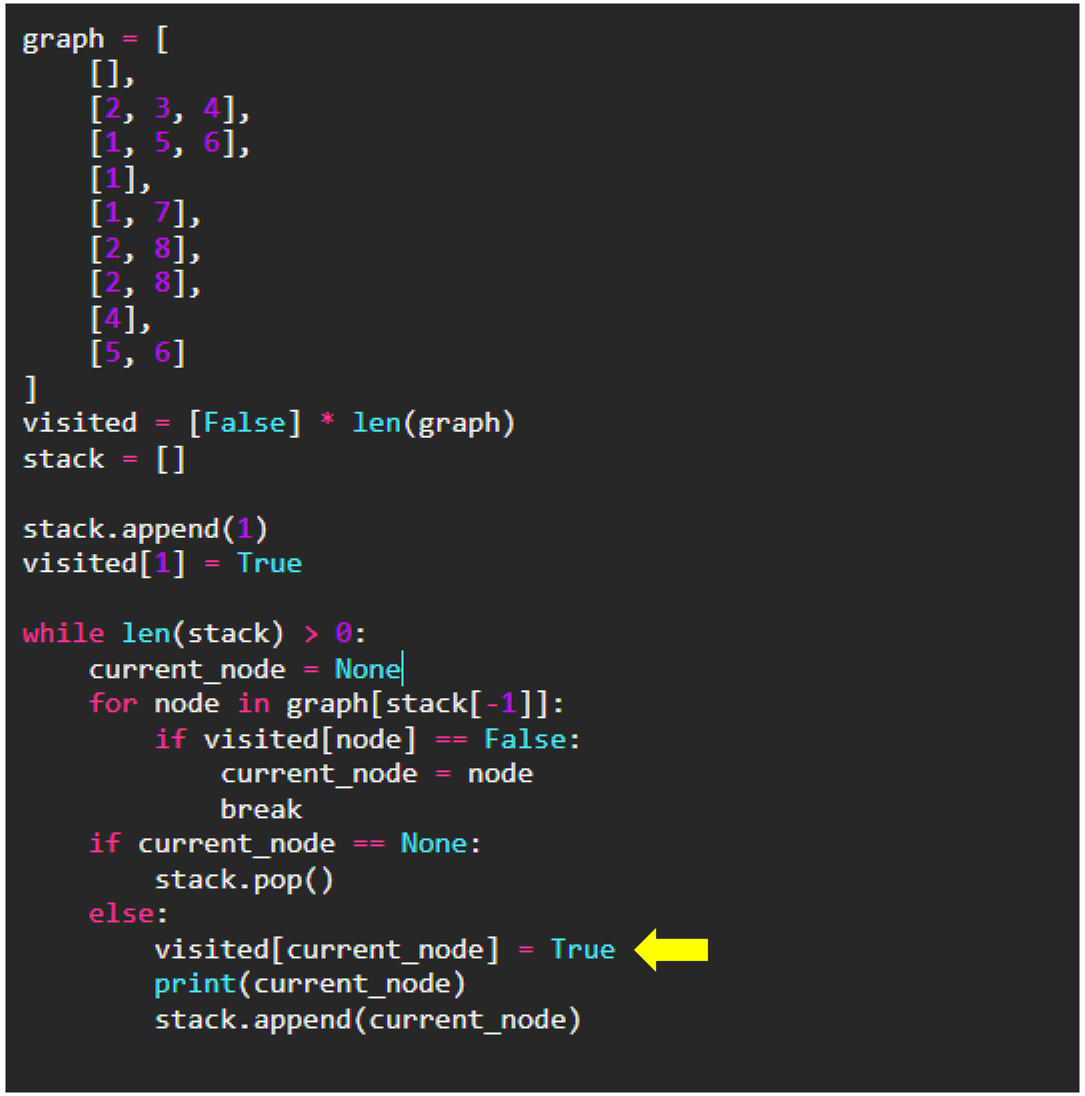
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]
visited = [False] * len(graph)
stack = []

stack.append(1)
visited[1] = True

while len(stack) > 0:
    current_node = None
    for node in graph[stack[-1]]:
        if visited[node] == False:
            current_node = node
            break
    if current_node == None:
        stack.pop()
    else:
        visited[current_node] = True
        print(current_node)
        stack.append(current_node)
```



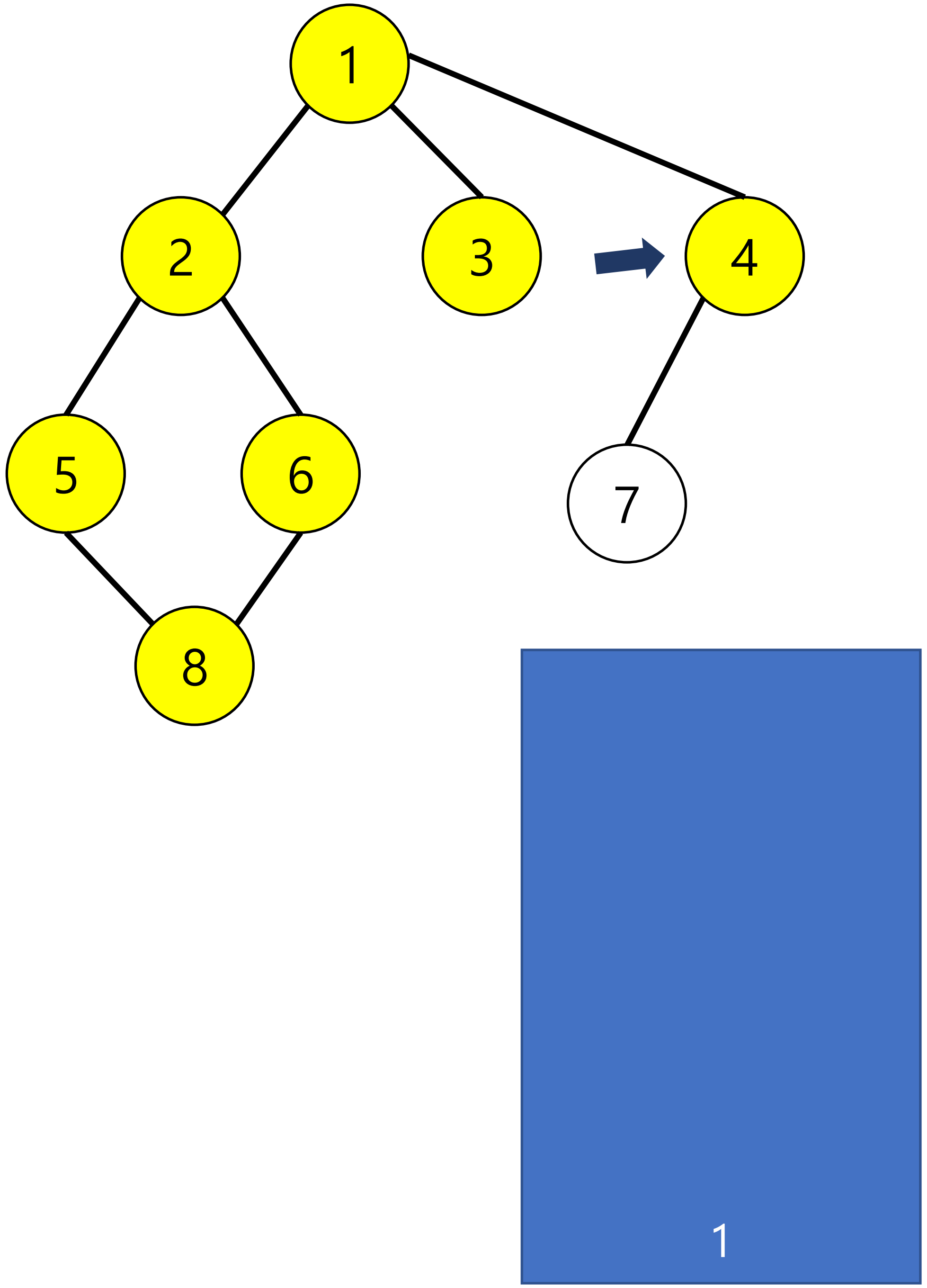


```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]
visited = [False] * len(graph)
stack = []

stack.append(1)
visited[1] = True

while len(stack) > 0:
    current_node = None
    for node in graph[stack[-1]]:
        if visited[node] == False:
            current_node = node
            break
    if current_node == None:
        stack.pop()
    else:
        visited[current_node] = True
        print(current_node)
        stack.append(current_node)
```

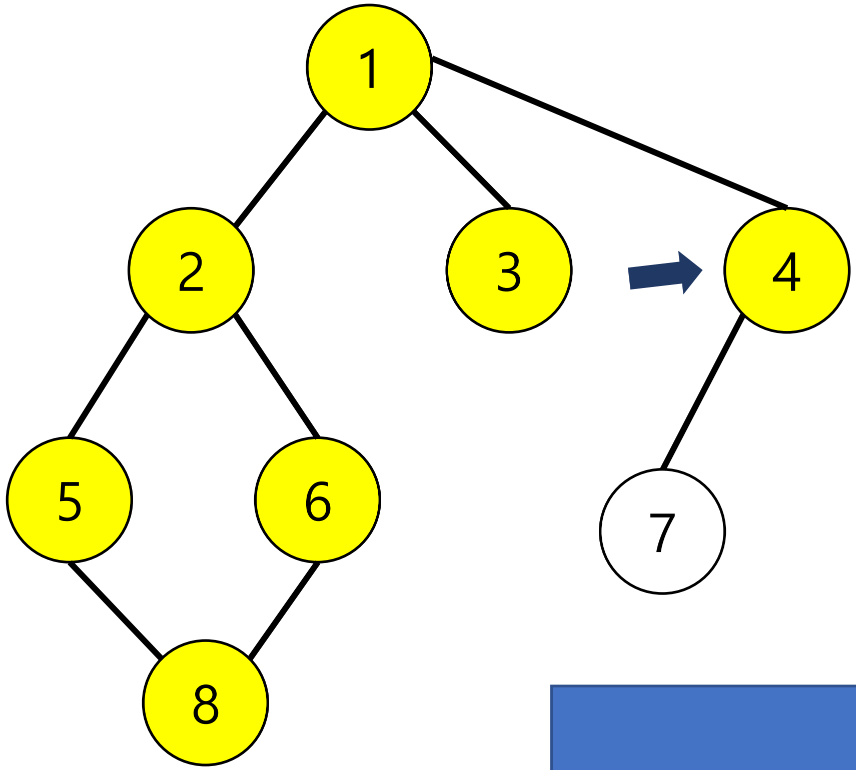


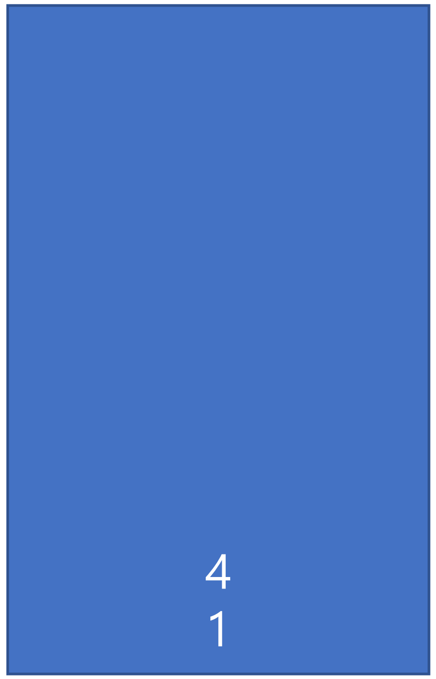


```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]
visited = [False] * len(graph)
stack = []

stack.append(1)
visited[1] = True

while len(stack) > 0:
    current_node = None
    for node in graph[stack[-1]]:
        if visited[node] == False:
            current_node = node
            break
    if current_node == None:
        stack.pop()
    else:
        visited[current_node] = True
        print(current_node)
        stack.append(current_node)
```

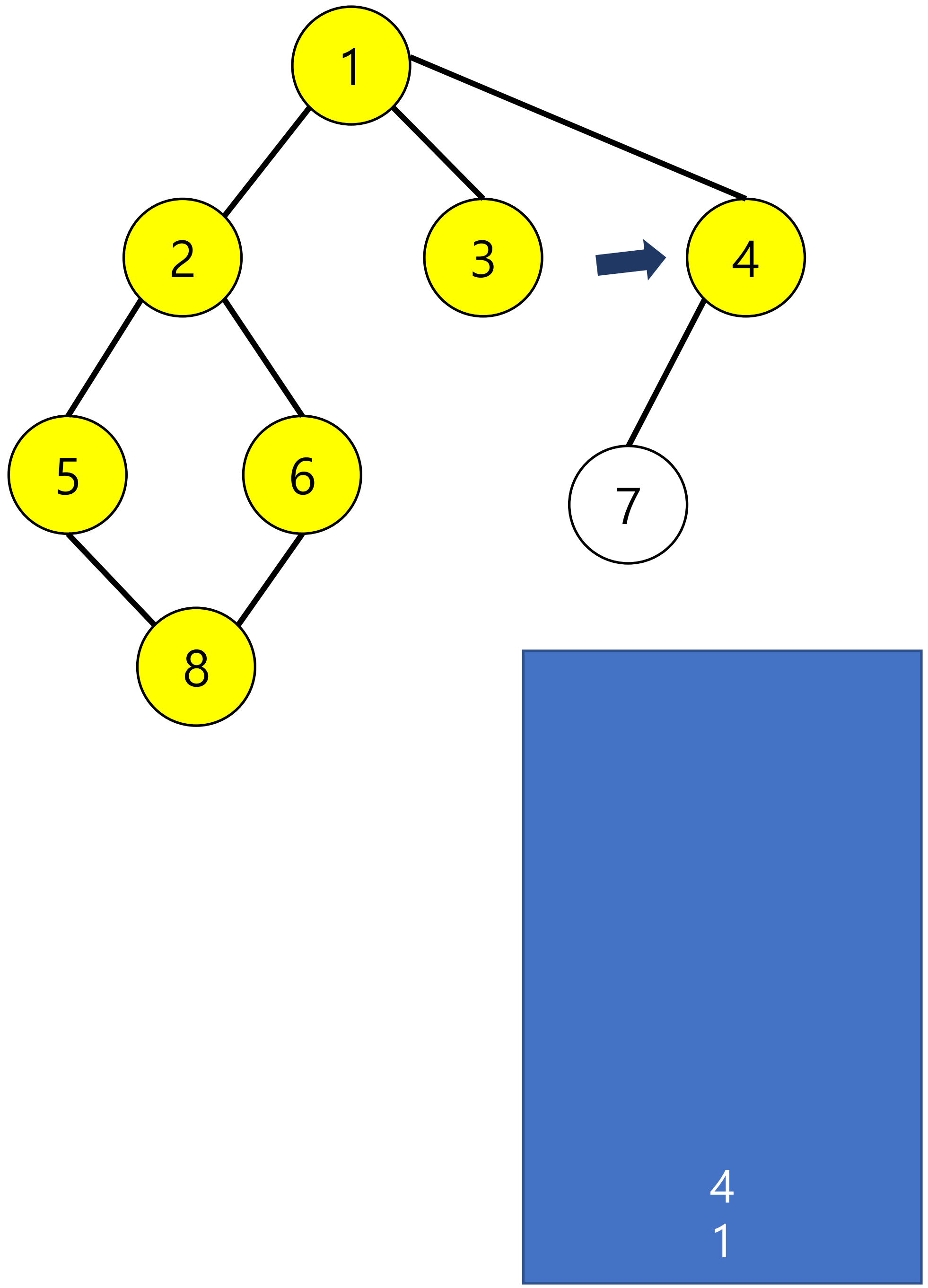


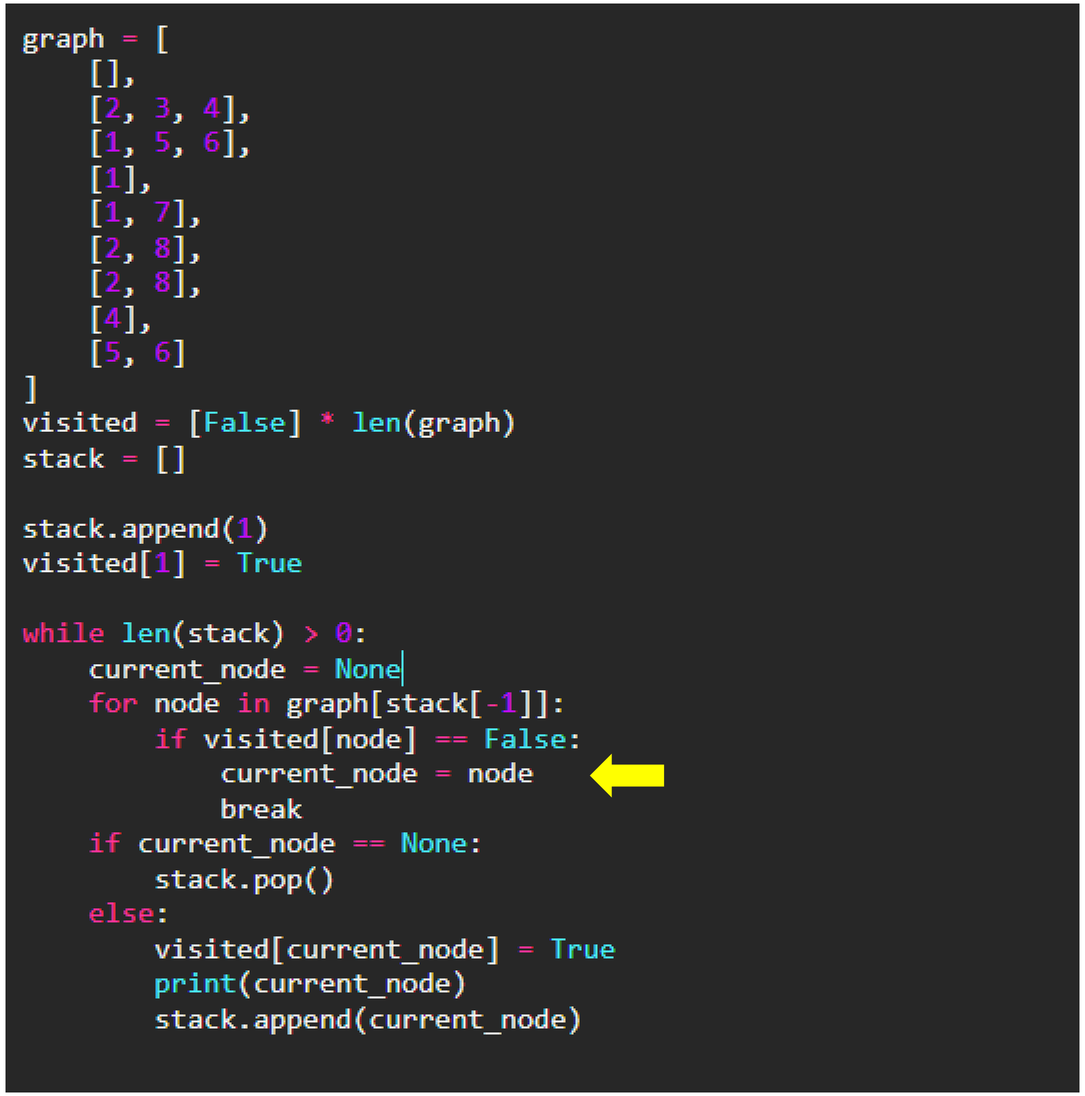
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]
visited = [False] * len(graph)
stack = []

stack.append(1)
visited[1] = True

while len(stack) > 0:
    current_node = None
    for node in graph[stack[-1]]:
        if visited[node] == False:
            current_node = node
            break
    if current_node == None:
        stack.pop()
    else:
        visited[current_node] = True
        print(current_node)
        stack.append(current_node)
```



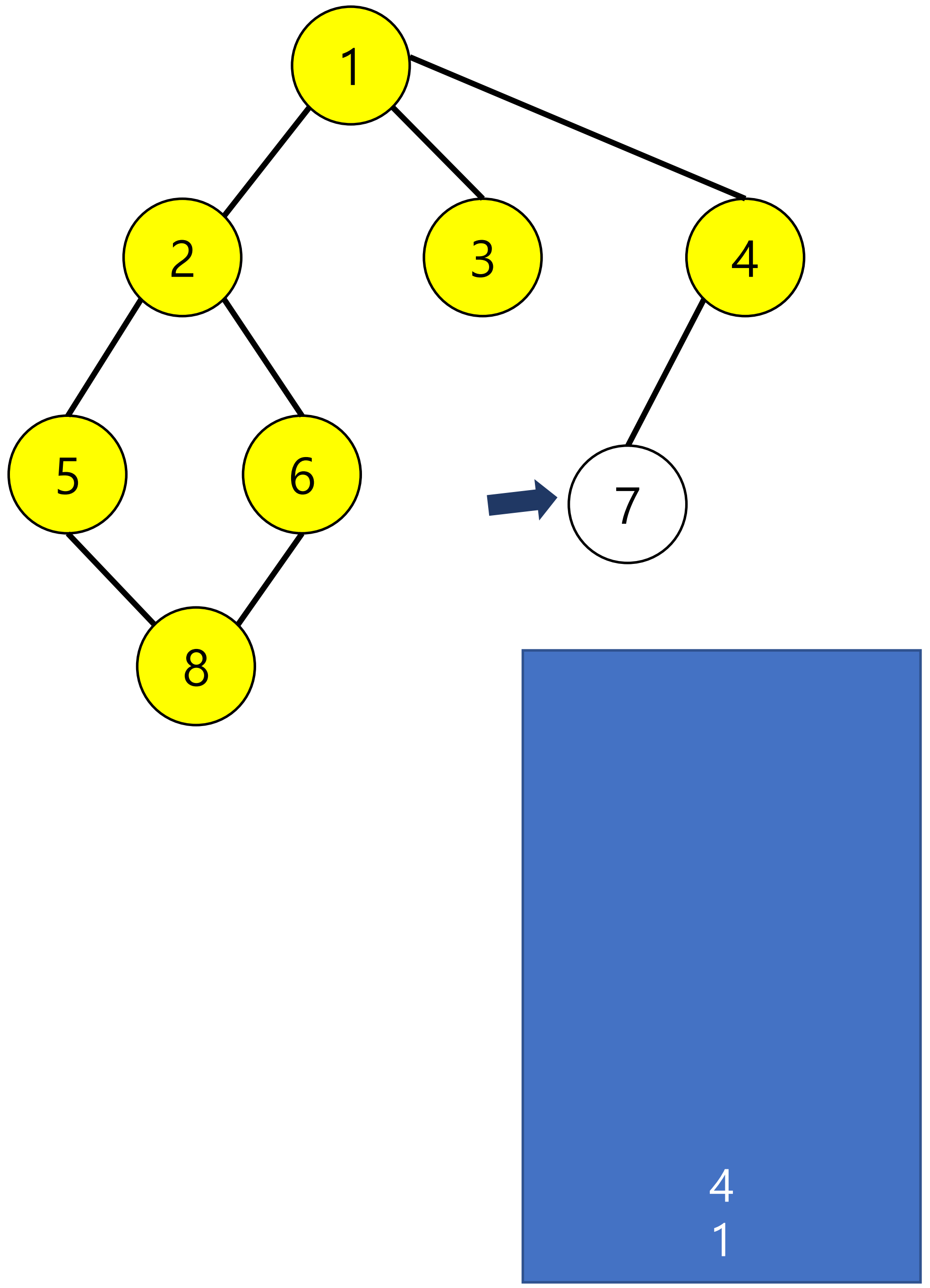


```python
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]
visited = [False] * len(graph)
stack = []

stack.append(1)
visited[1] = True

while len(stack) > 0:
    current_node = None
    for node in graph[stack[-1]]:
        if visited[node] == False:
            current_node = node
            break
    if current_node == None:
        stack.pop()
    else:
        visited[current_node] = True
        print(current_node)
        stack.append(current_node)
```

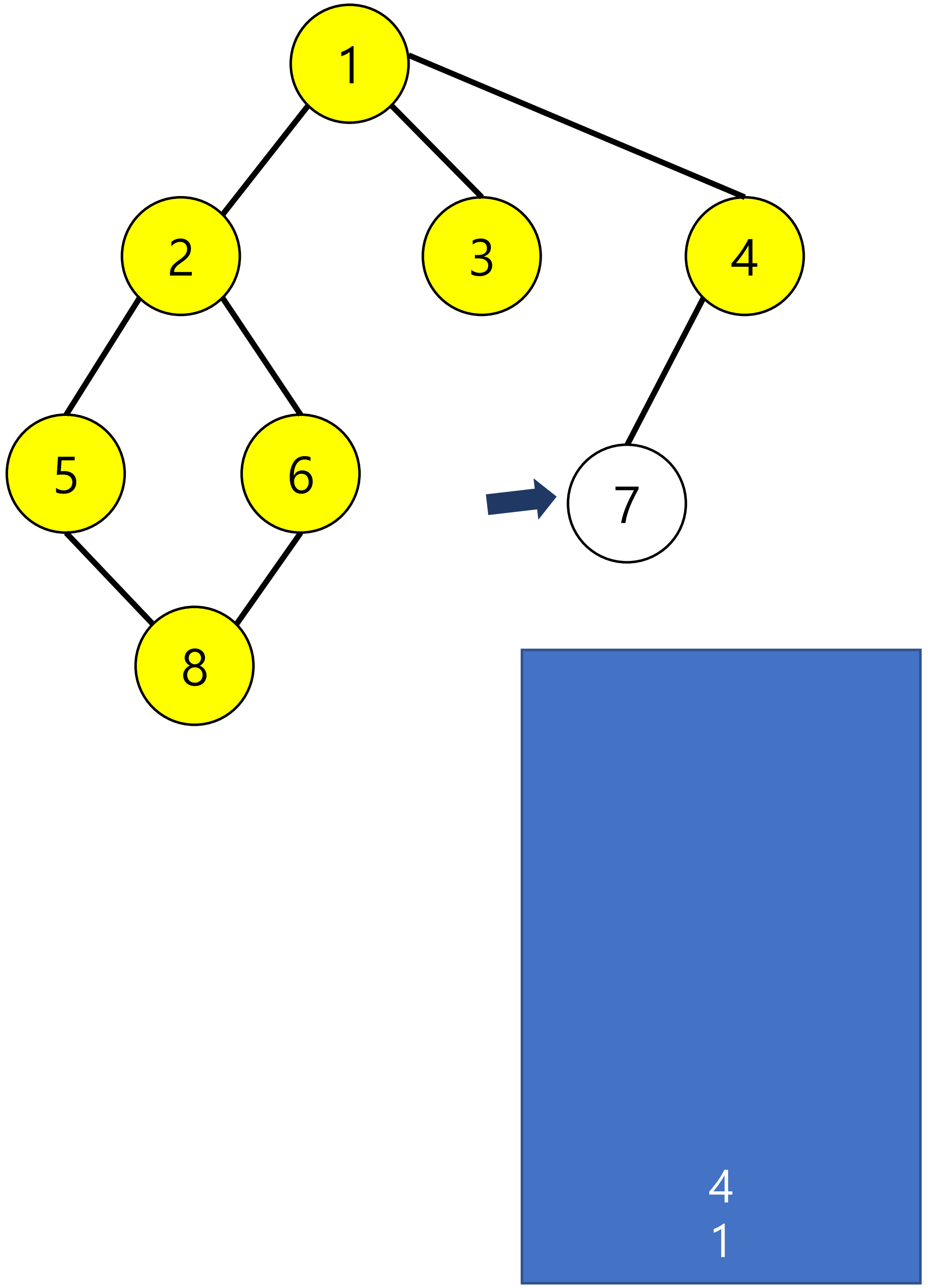


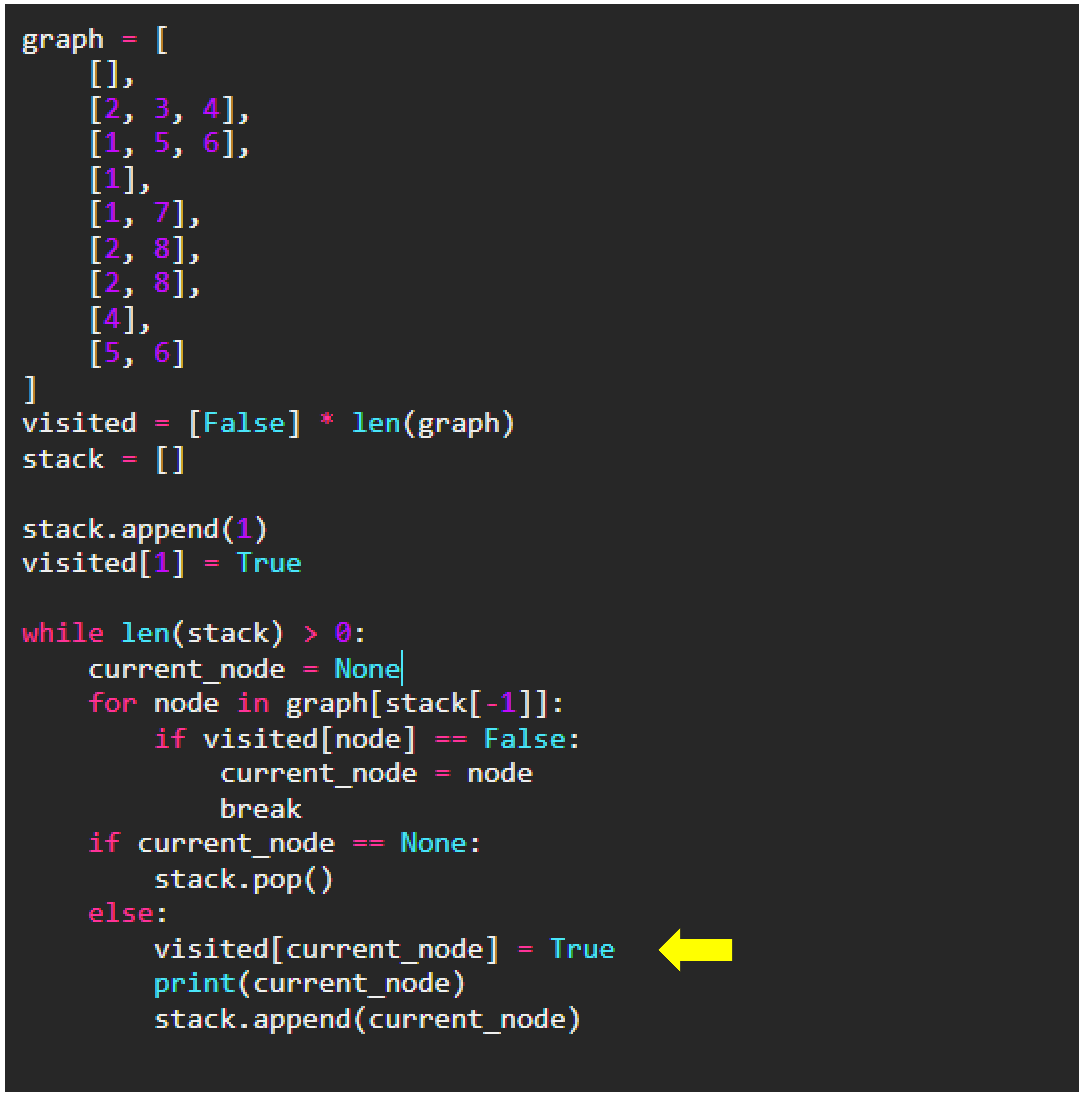


```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]
visited = [False] * len(graph)
stack = []

stack.append(1)
visited[1] = True

while len(stack) > 0:
    current_node = None
    for node in graph[stack[-1]]:
        if visited[node] == False:
            current_node = node
            break
    if current_node == None:
        stack.pop()
    else:
        visited[current_node] = True
        print(current_node)
        stack.append(current_node)
```

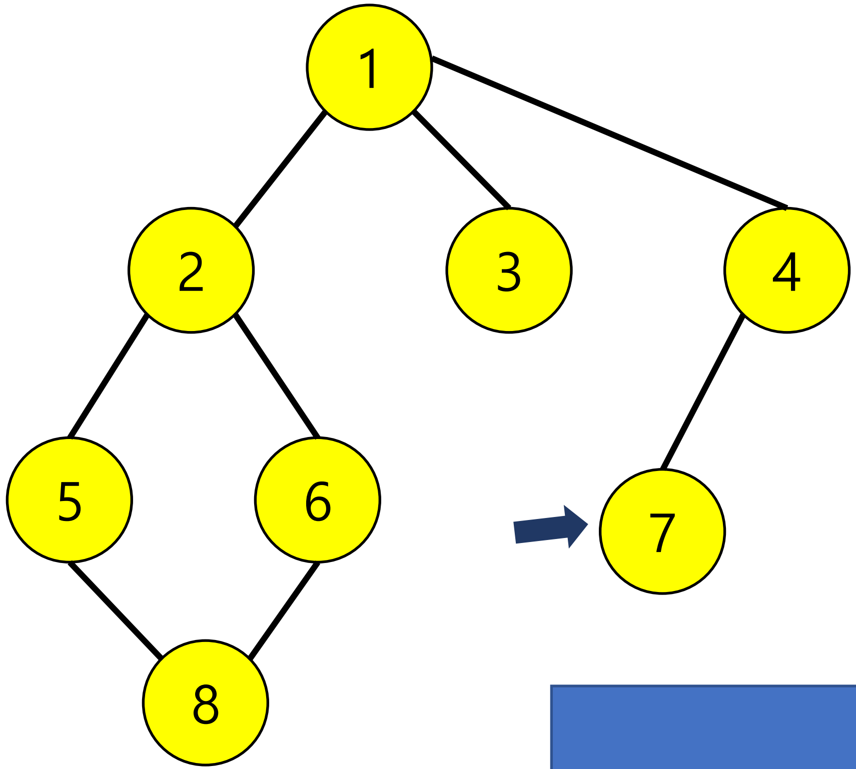


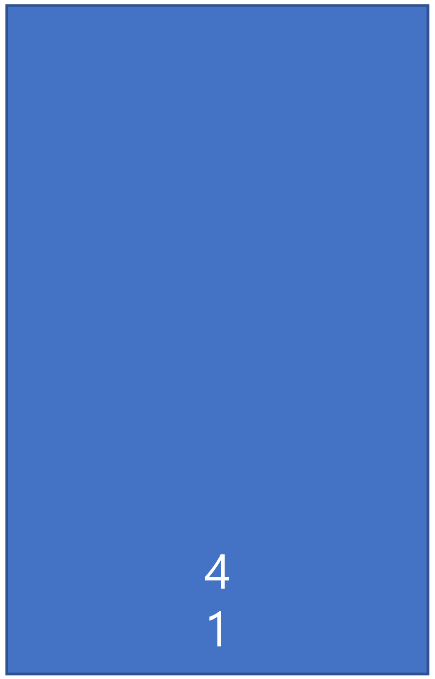
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]
visited = [False] * len(graph)
stack = []

stack.append(1)
visited[1] = True

while len(stack) > 0:
    current_node = None
    for node in graph[stack[-1]]:
        if visited[node] == False:
            current_node = node
            break
    if current_node == None:
        stack.pop()
    else:
        visited[current_node] = True
        print(current_node)
        stack.append(current_node)
```

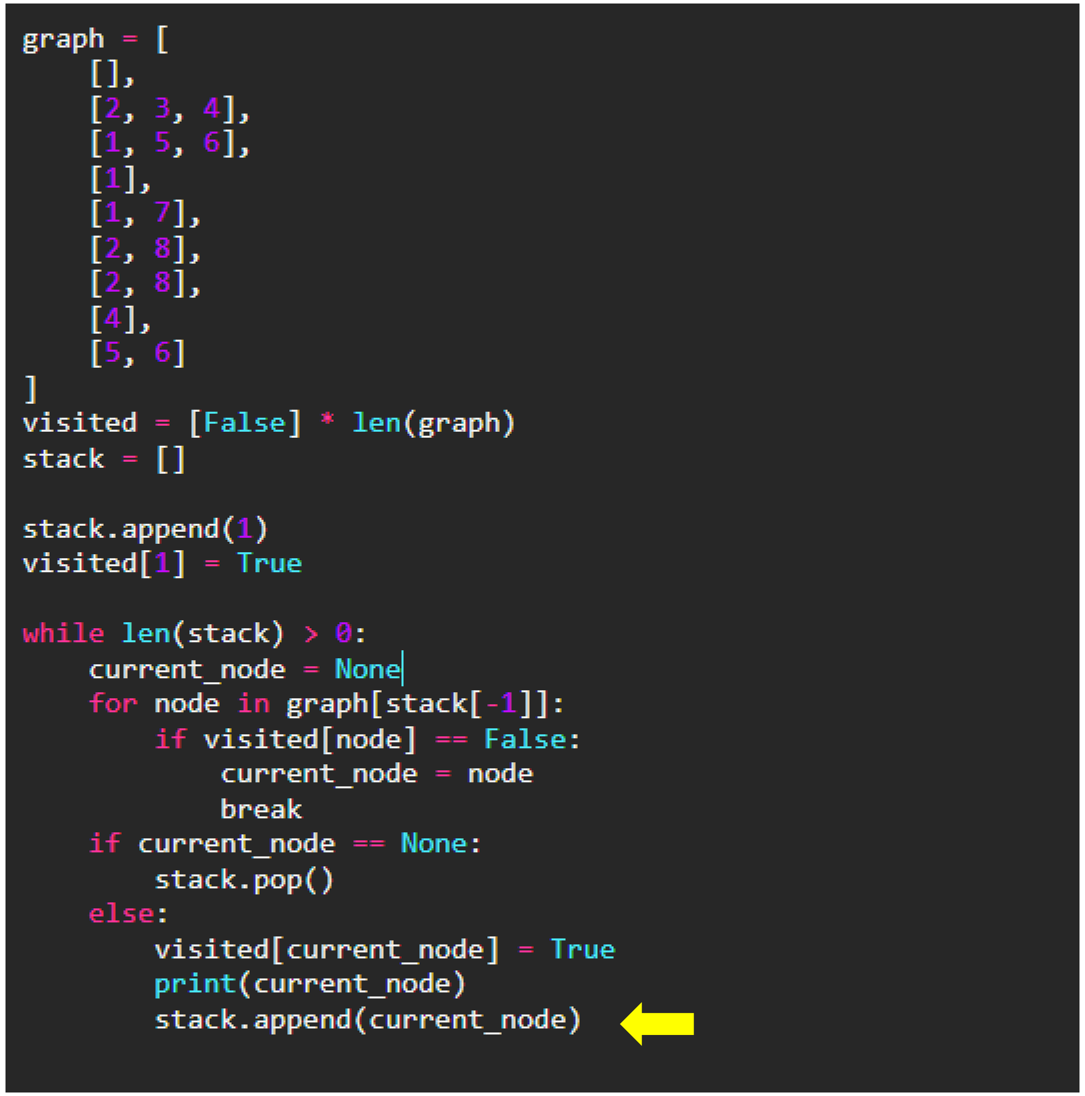




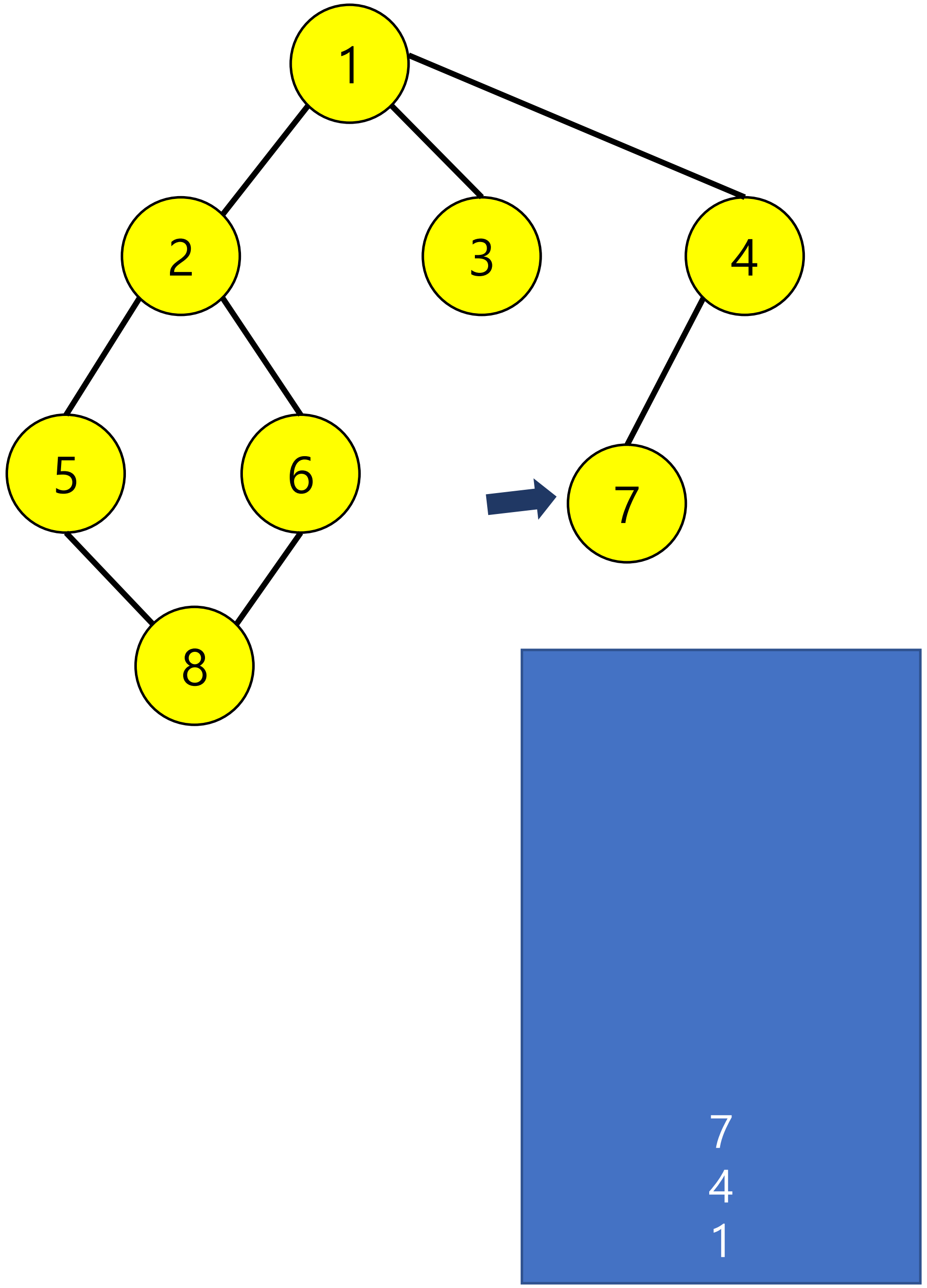
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]
visited = [False] * len(graph)
stack = []

stack.append(1)
visited[1] = True

while len(stack) > 0:
    current_node = None
    for node in graph[stack[-1]]:
        if visited[node] == False:
            current_node = node
            break
    if current_node == None:
        stack.pop()
    else:
        visited[current_node] = True
        print(current_node)
        stack.append(current_node)
```

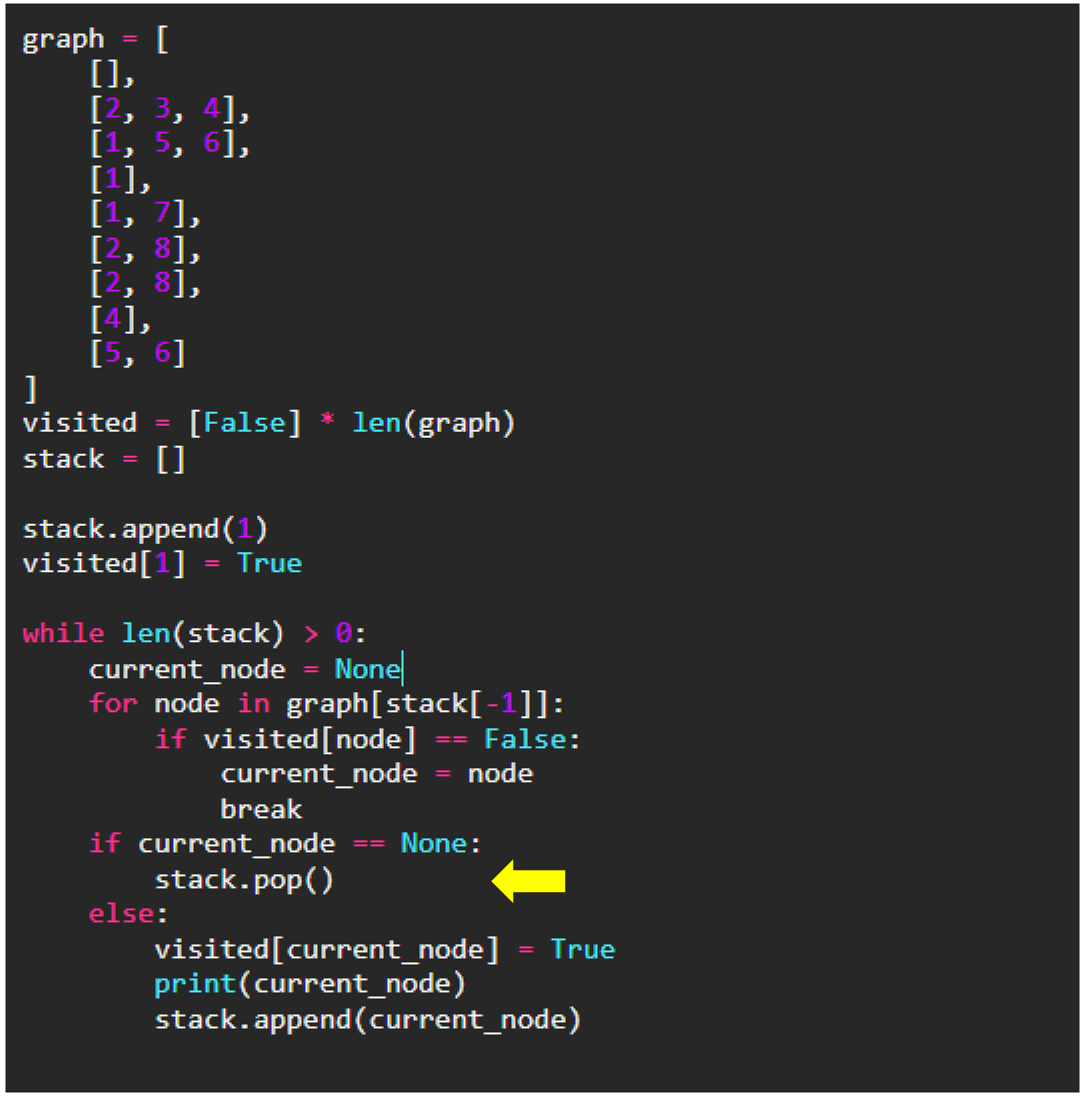




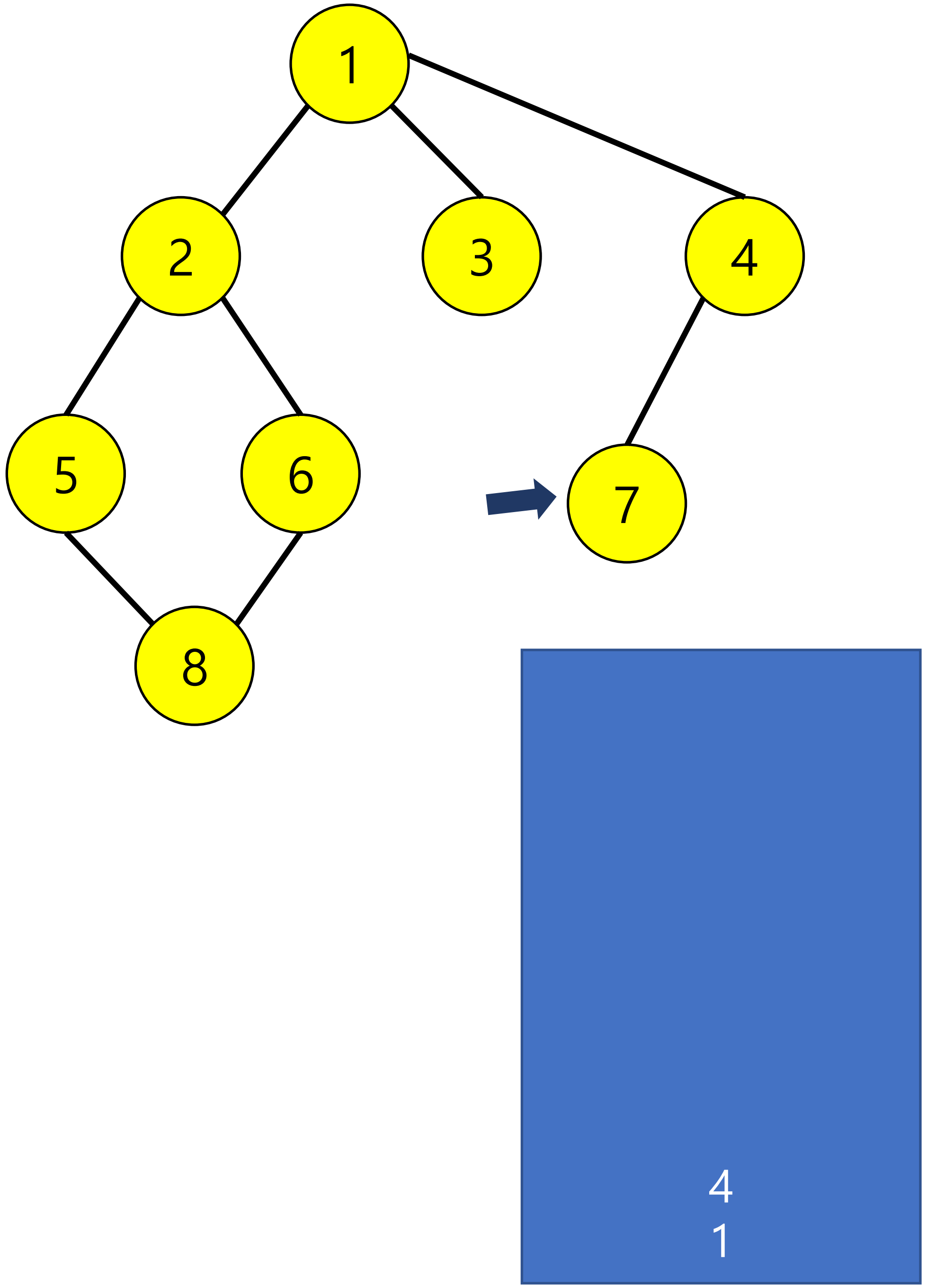


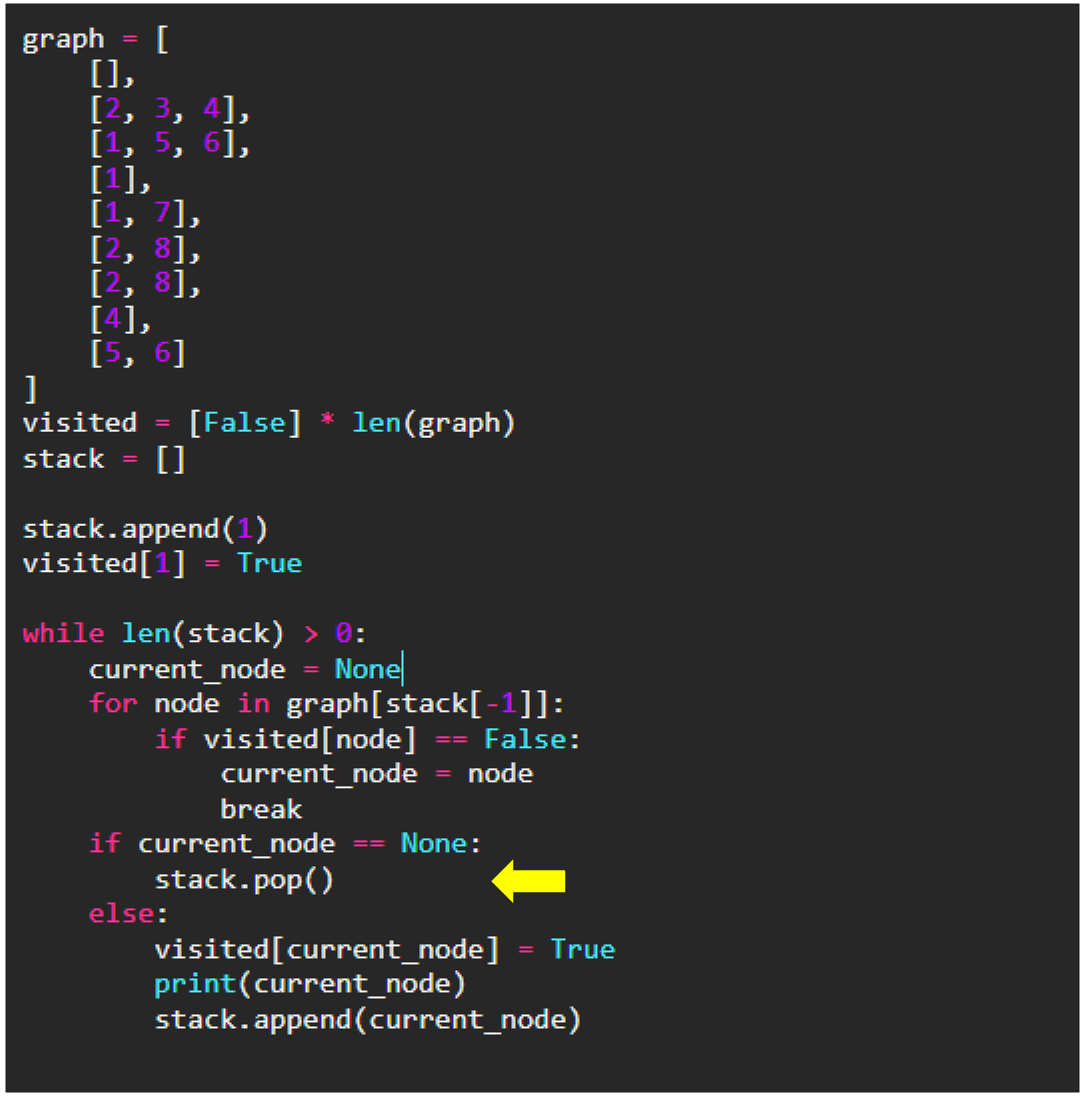
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]
visited = [False] * len(graph)
stack = []

stack.append(1)
visited[1] = True

while len(stack) > 0:
    current_node = None
    for node in graph[stack[-1]]:
        if visited[node] == False:
            current_node = node
            break
    if current_node == None:
        stack.pop()
    else:
        visited[current_node] = True
        print(current_node)
        stack.append(current_node)
```



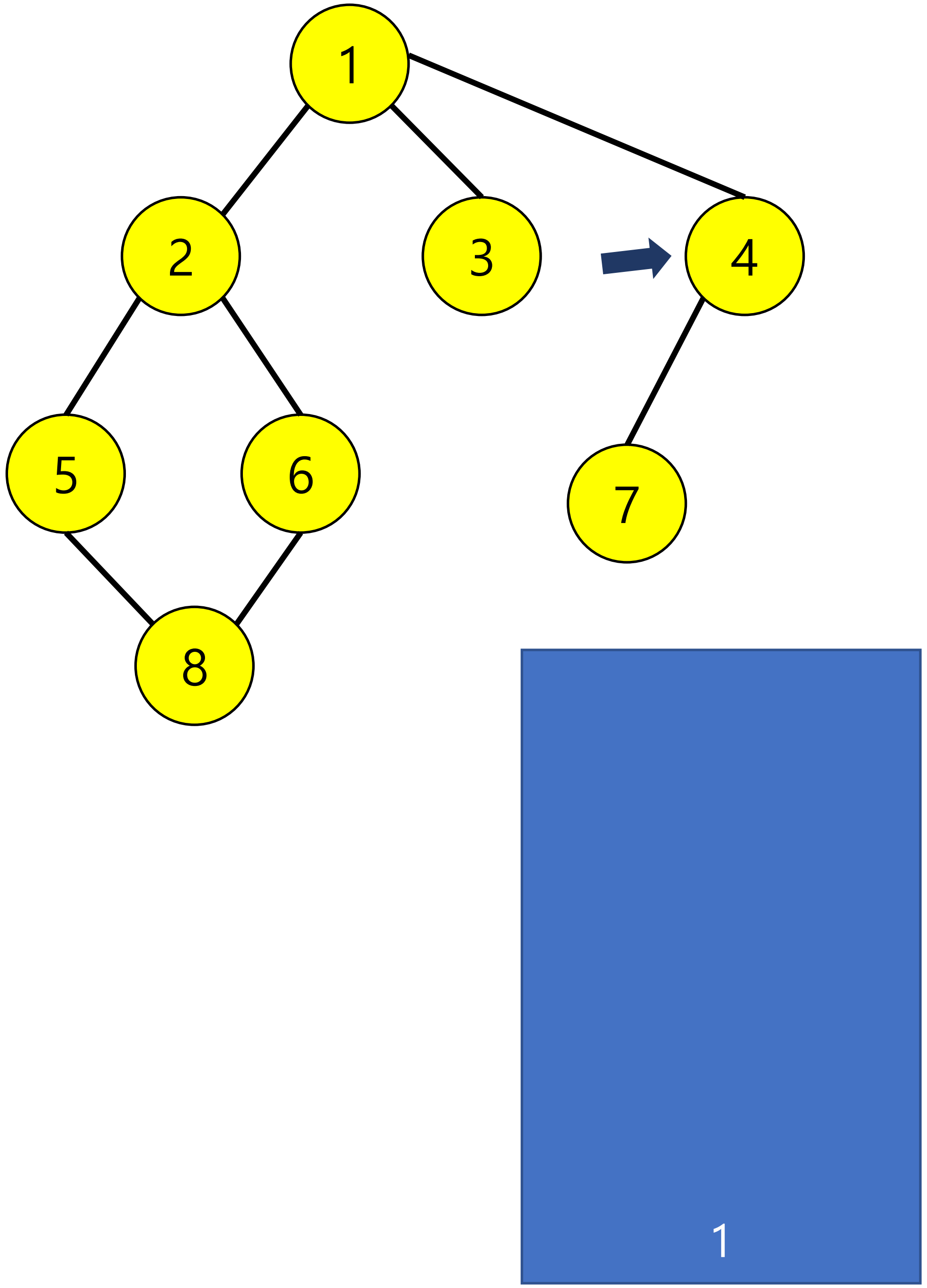


```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]
visited = [False] * len(graph)
stack = []

stack.append(1)
visited[1] = True

while len(stack) > 0:
    current_node = None
    for node in graph[stack[-1]]:
        if visited[node] == False:
            current_node = node
            break
    if current_node == None:
        stack.pop()
    else:
        visited[current_node] = True
        print(current_node)
        stack.append(current_node)
```

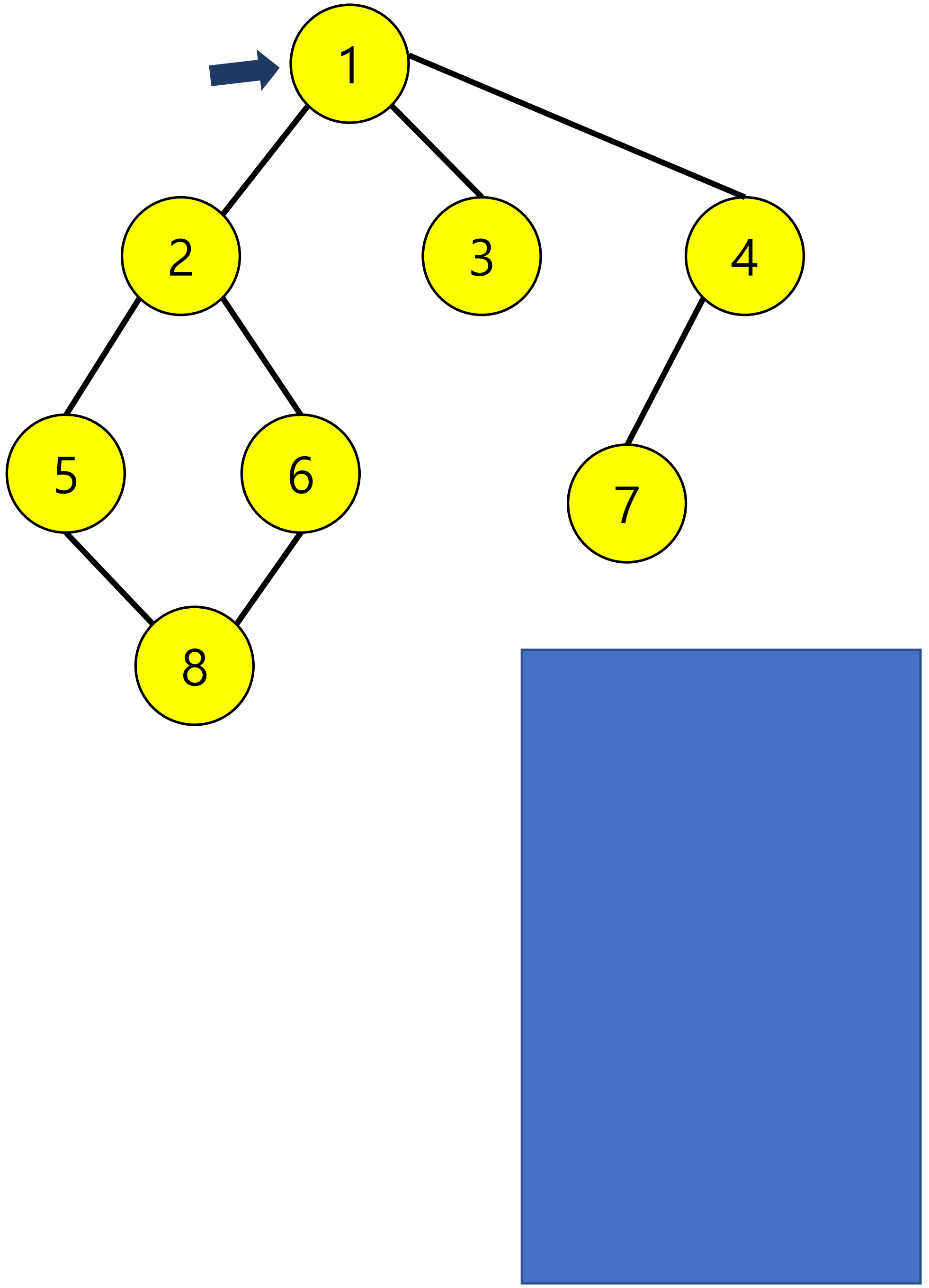




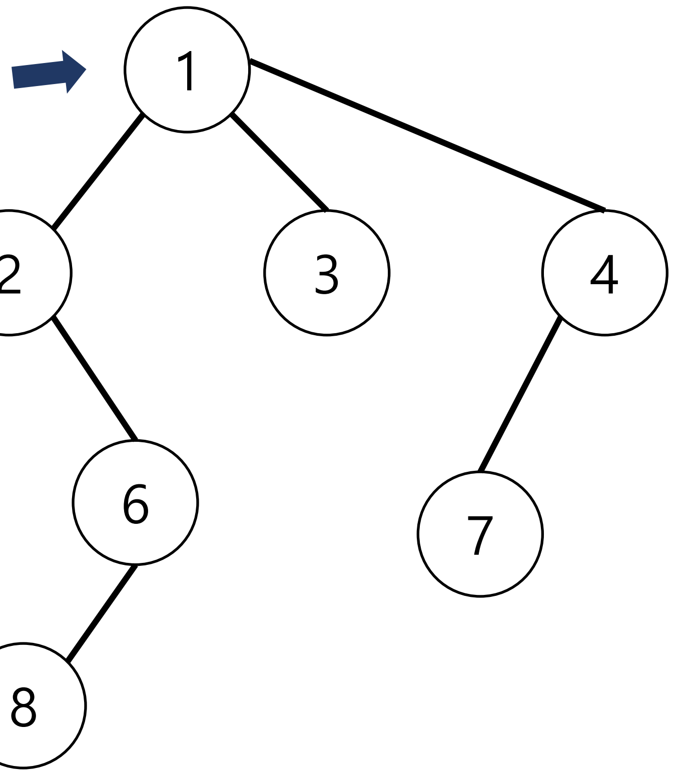
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]
visited = [False] * len(graph)
stack = []

stack.append(1)
visited[1] = True

while len(stack) > 0:
    current_node = None
    for node in graph[stack[-1]]:
        if visited[node] == False:
            current_node = node
            break
    if current_node == None:
        stack.pop()
    else:
        visited[current_node] = True
        print(current_node)
        stack.append(current_node)
```

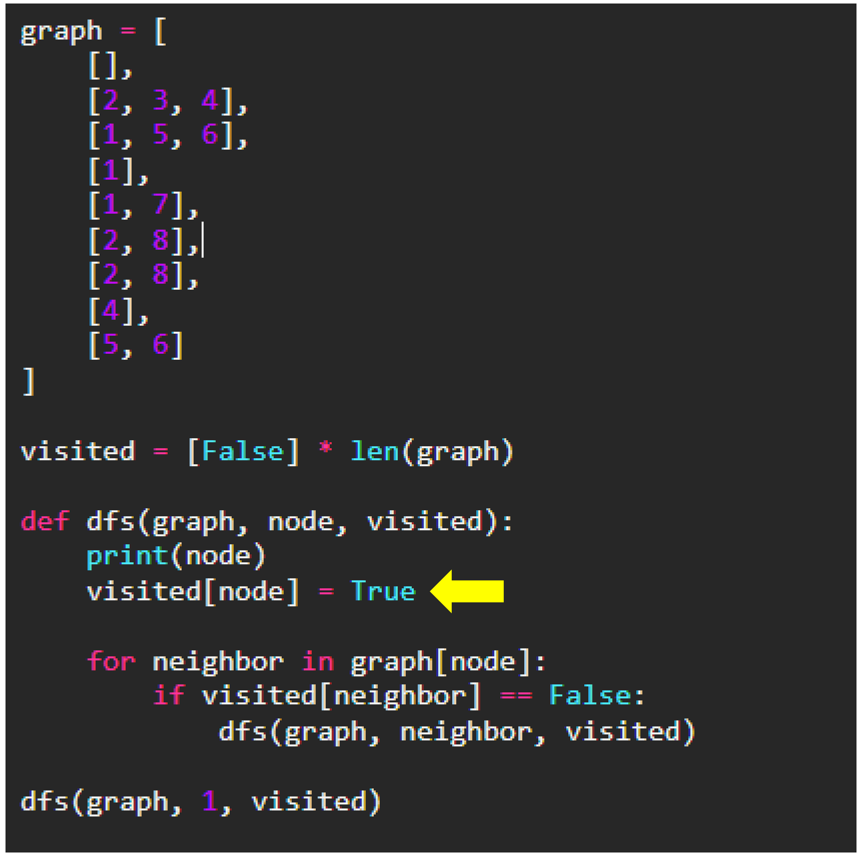


```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]
visited = [False] * len(graph)
stack = []

stack.append(1)
visited[1] = True

while len(stack) > 0:
    current_node = None
    for node in graph[stack[-1]]:
        if visited[node] == False:
            current_node = node
            break
    if current_node == None:
        stack.pop()
    else:
        visited[current_node] = True
        print(current_node)
        stack.append(current_node)
```

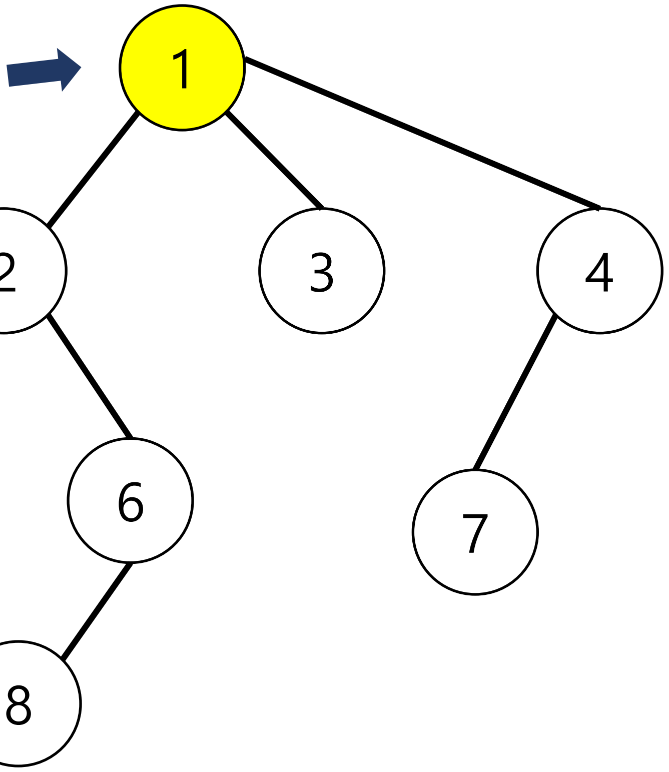


```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

visited = [False] * len(graph)

def dfs(graph, node, visited):
    print(node)
    visited[node] = True

    for neighbor in graph[node]:
        if visited[neighbor] == False:
            dfs(graph, neighbor, visited)

dfs(graph, 1, visited)
```



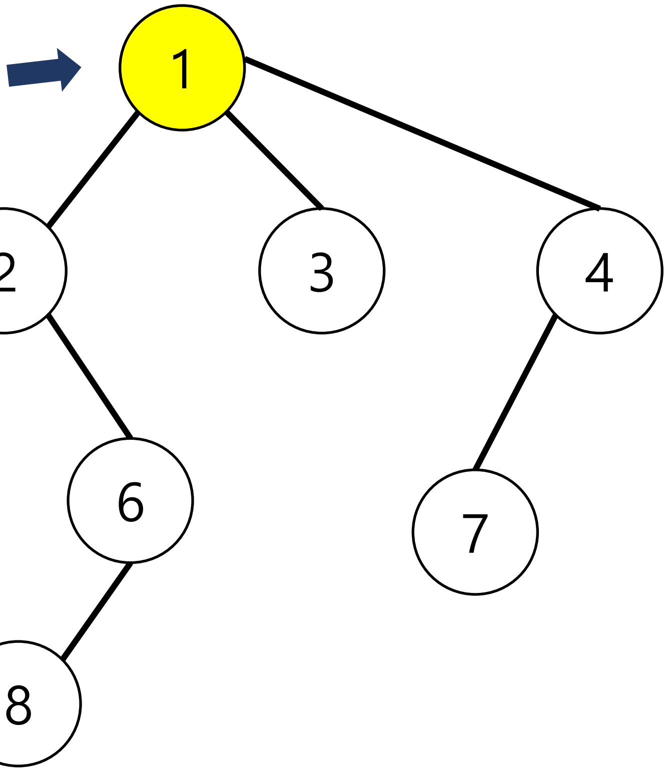


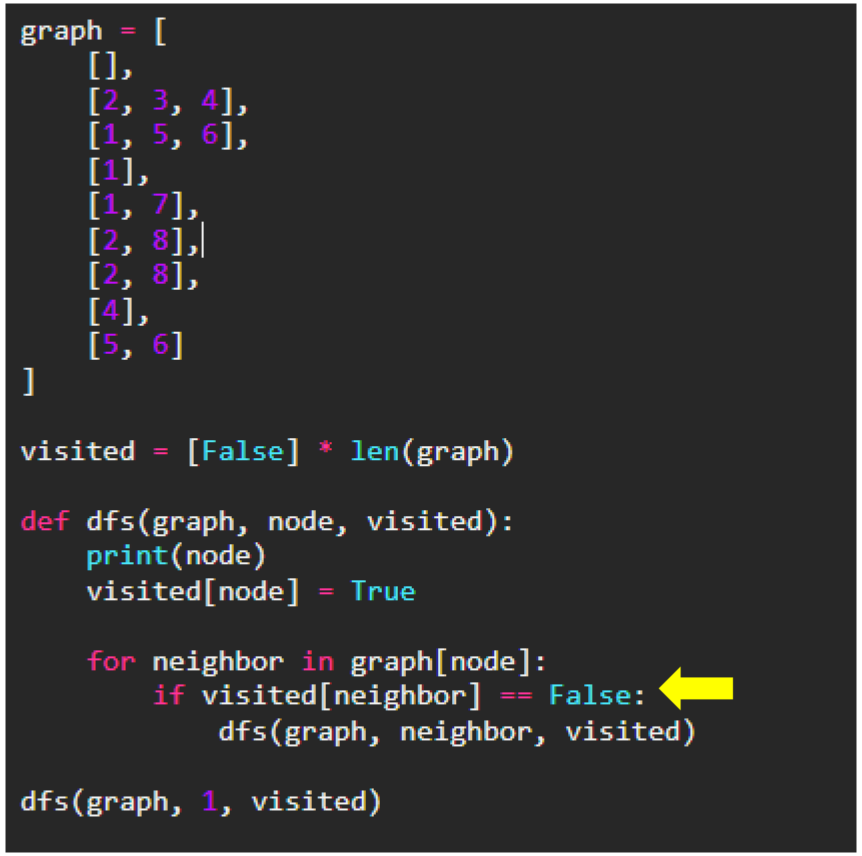
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

visited = [False] * len(graph)

def dfs(graph, node, visited):
    print(node)
    visited[node] = True

    for neighbor in graph[node]:
        if visited[neighbor] == False:
            dfs(graph, neighbor, visited)

dfs(graph, 1, visited)
```



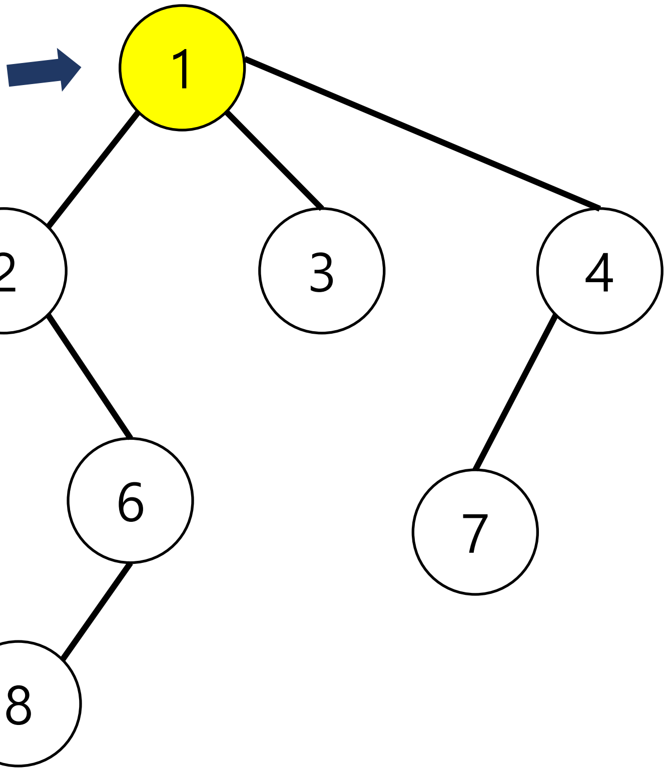
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

visited = [False] * len(graph)

def dfs(graph, node, visited):
    print(node)
    visited[node] = True

    for neighbor in graph[node]:
        if visited[neighbor] == False:
            dfs(graph, neighbor, visited)

dfs(graph, 1, visited)
```

```
for(i=0;i<graph[node].length;i++)
    neighbor = graph[node][i];
```

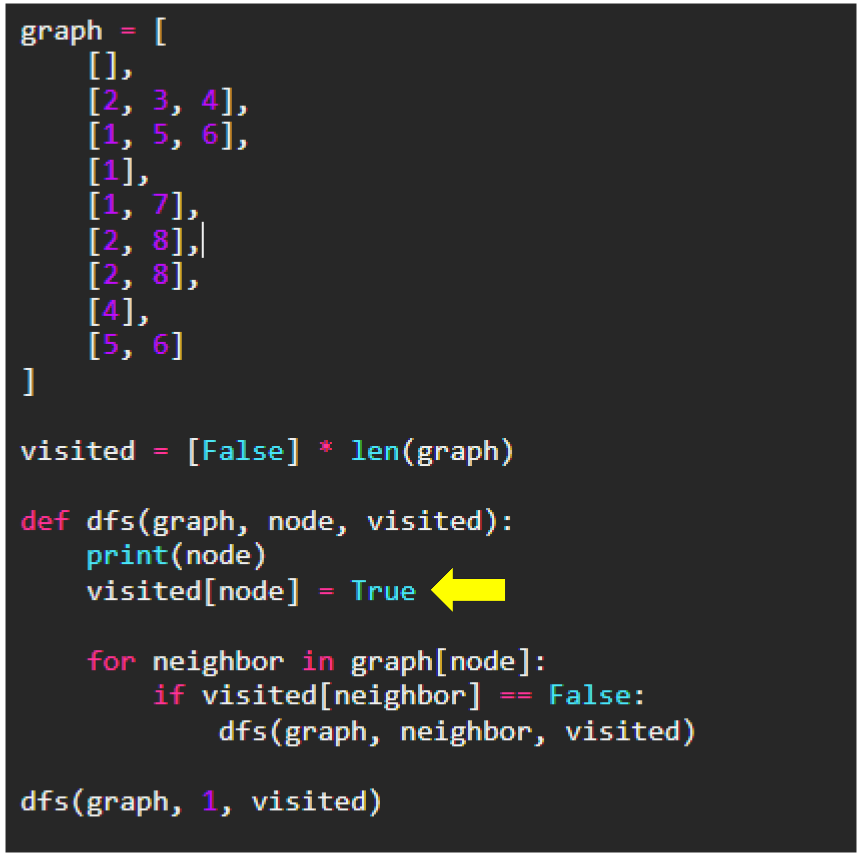




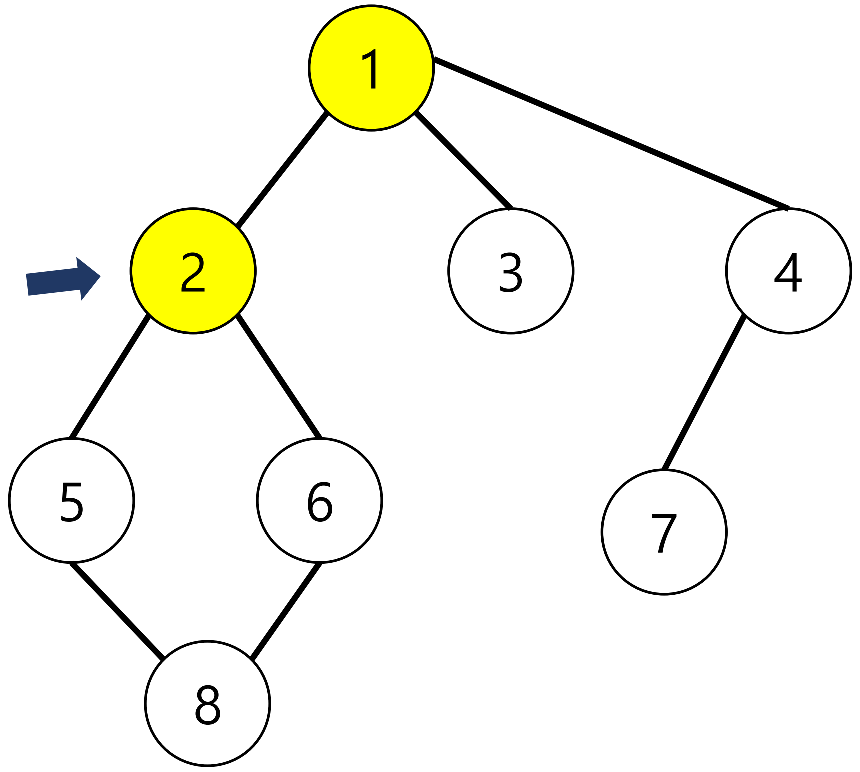
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

visited = [False] * len(graph)

def dfs(graph, node, visited):
    print(node)
    visited[node] = True

    for neighbor in graph[node]:
        if visited[neighbor] == False:
            dfs(graph, neighbor, visited)

dfs(graph, 1, visited)
```



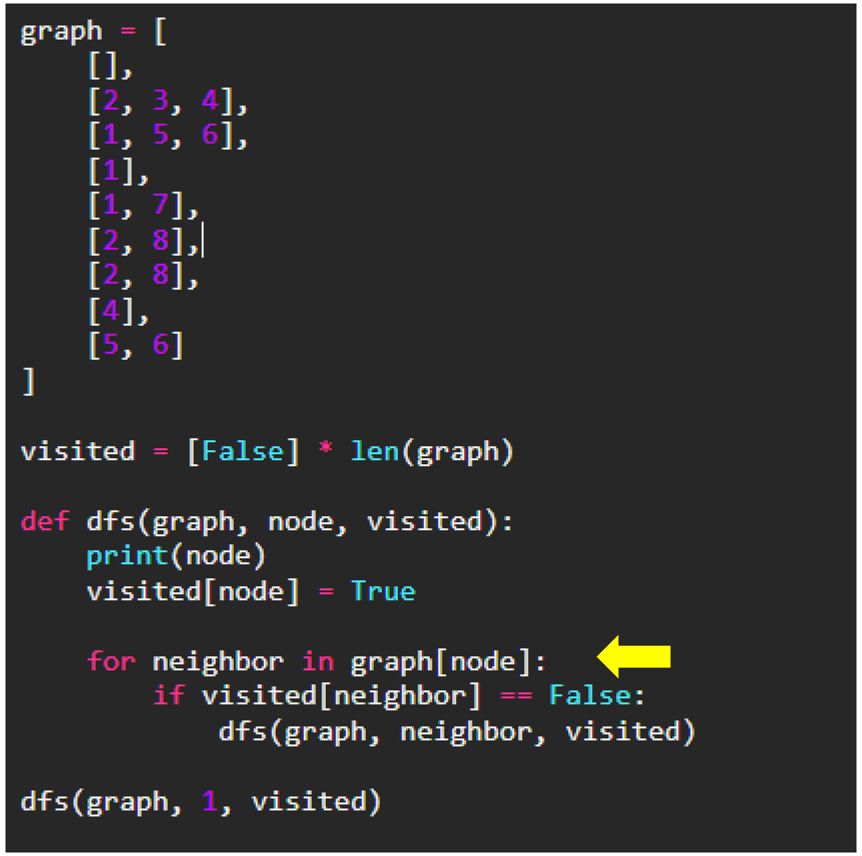
neighbor = 3
neighbor = 4



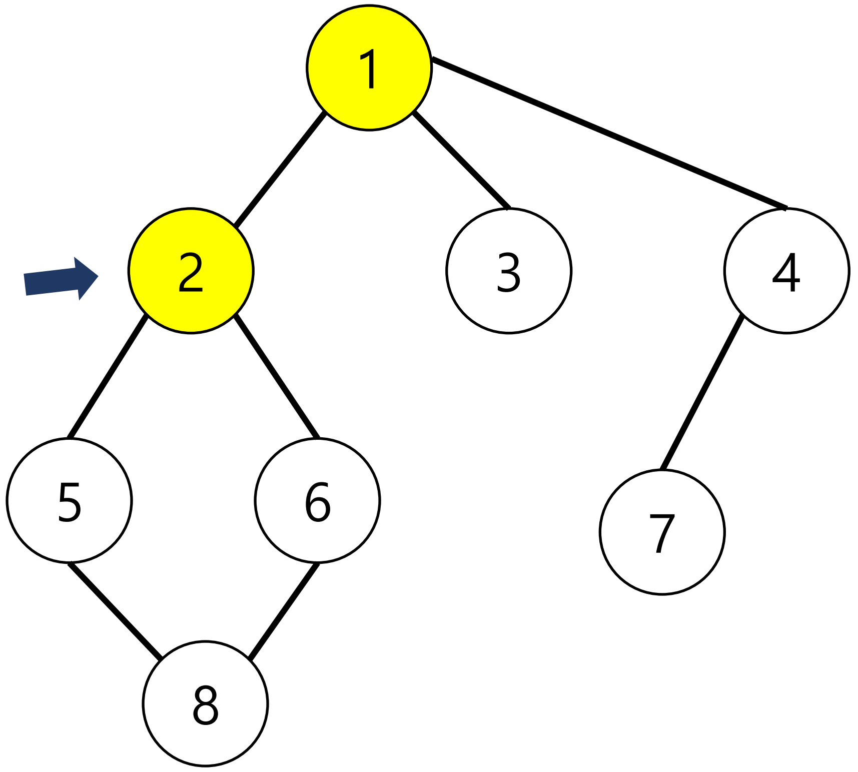
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

visited = [False] * len(graph)

def dfs(graph, node, visited):
    print(node)
    visited[node] = True

    for neighbor in graph[node]:
        if visited[neighbor] == False:
            dfs(graph, neighbor, visited)

dfs(graph, 1, visited)
```



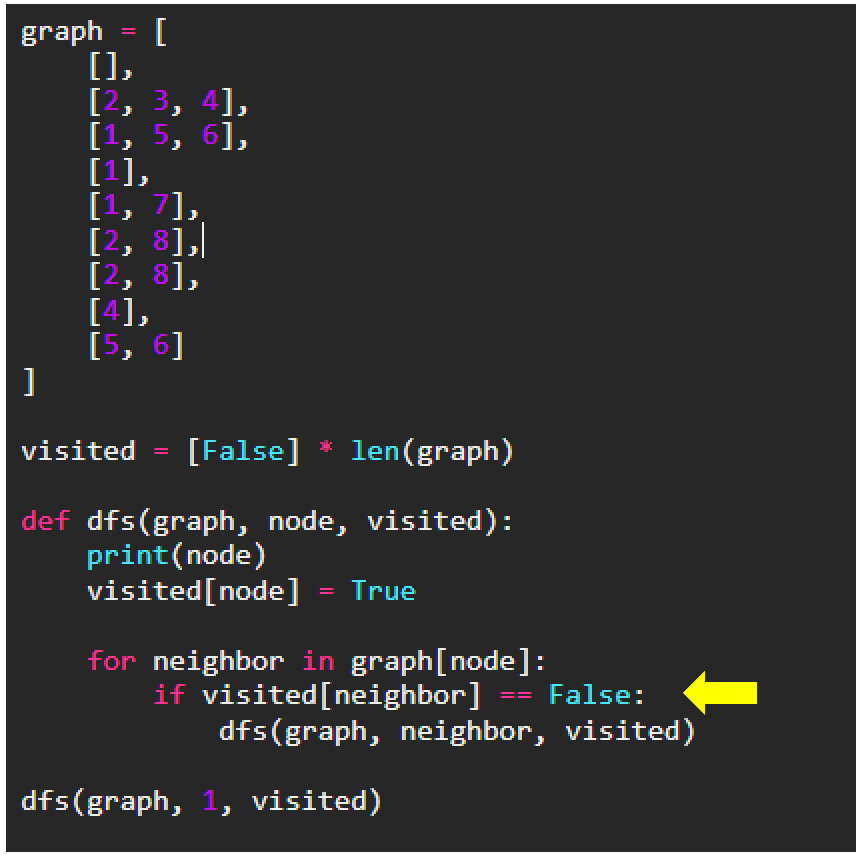
neighbor = 3
neighbor = 4



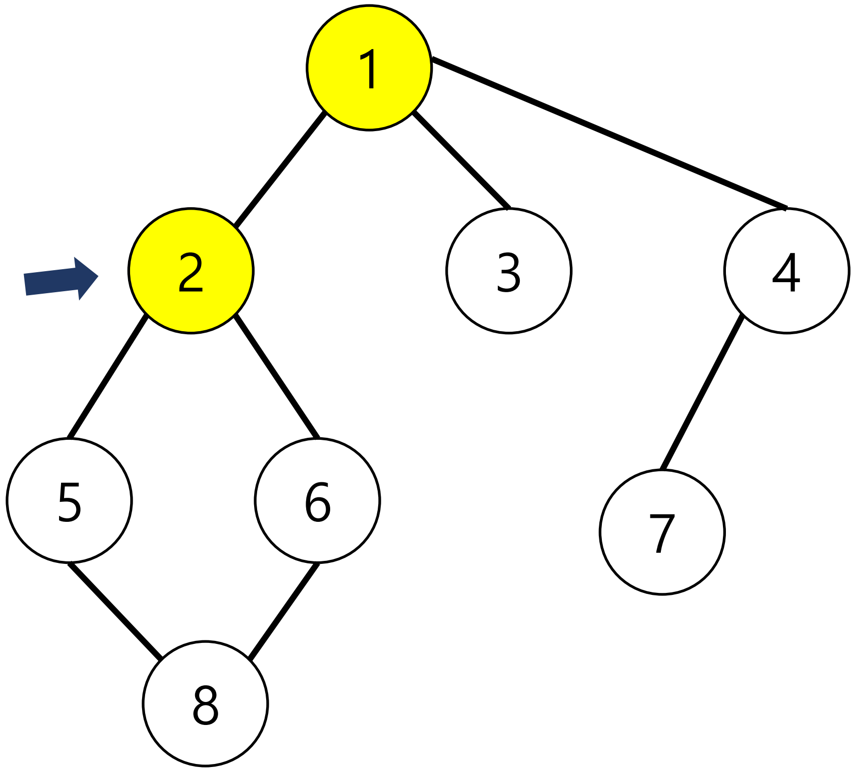
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

visited = [False] * len(graph)

def dfs(graph, node, visited):
    print(node)
    visited[node] = True

    for neighbor in graph[node]:
        if visited[neighbor] == False:
            dfs(graph, neighbor, visited)

dfs(graph, 1, visited)
```



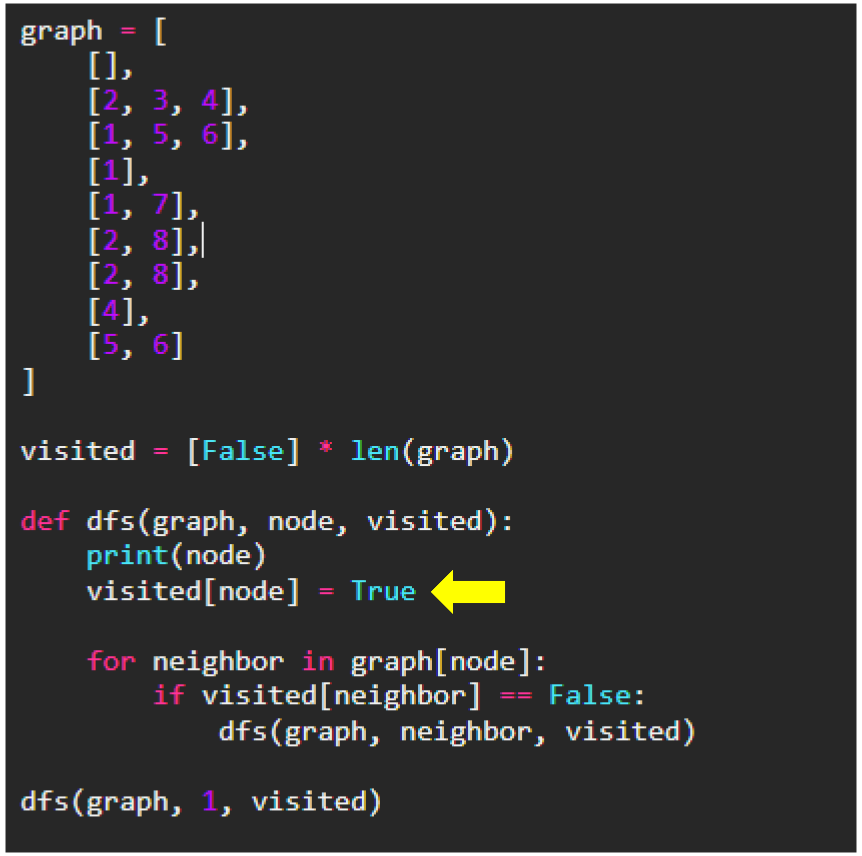
neighbor = 3
neighbor = 4



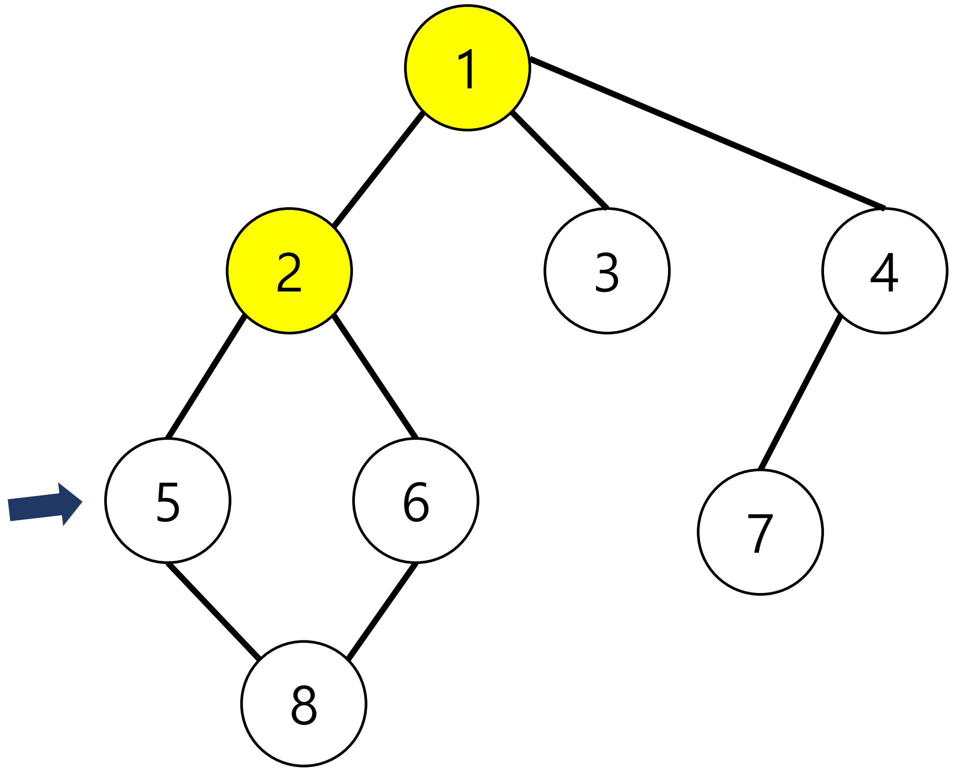
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

visited = [False] * len(graph)

def dfs(graph, node, visited):
    print(node)
    visited[node] = True

    for neighbor in graph[node]:
        if visited[neighbor] == False:
            dfs(graph, neighbor, visited)

dfs(graph, 1, visited)
```



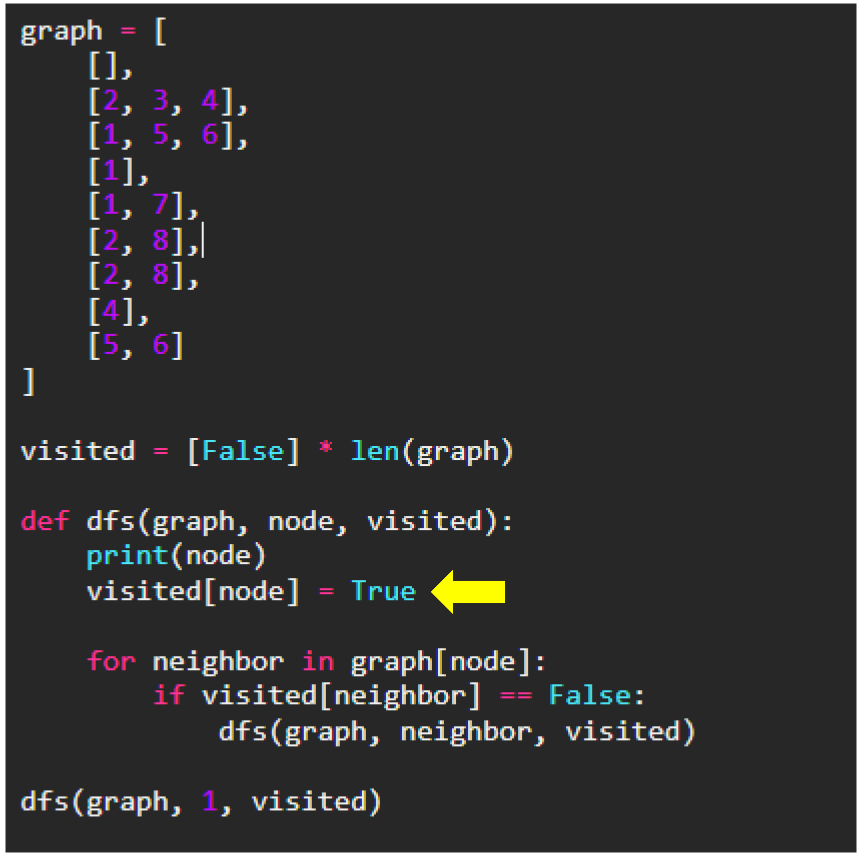
neighbor = 3
neighbor = 4
neighbor = 6



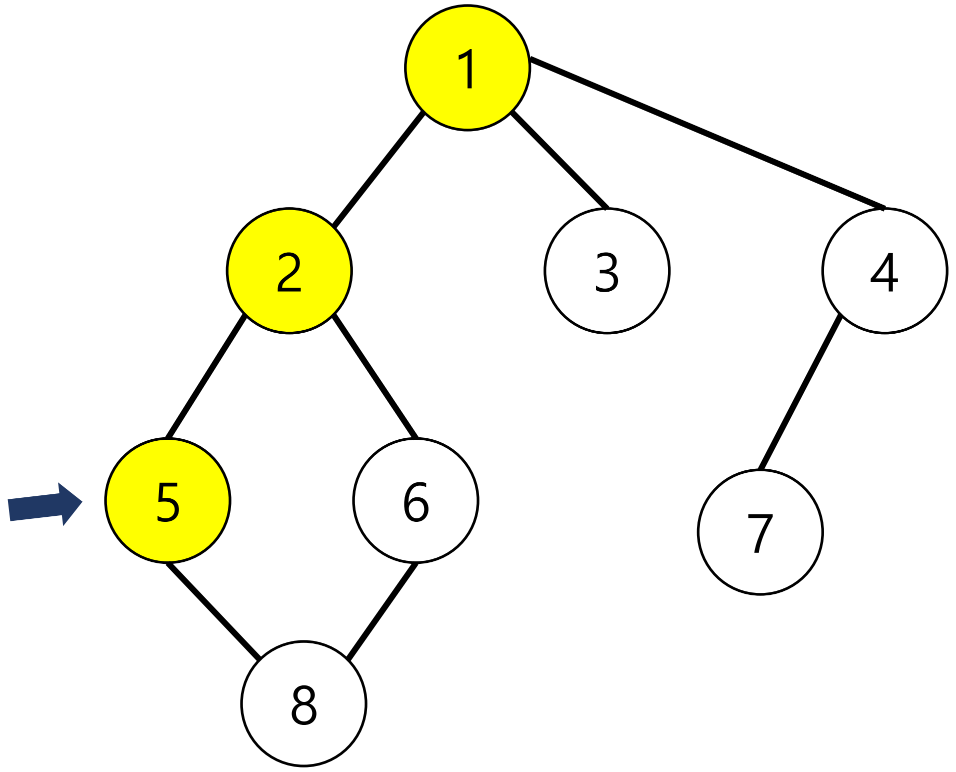
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

visited = [False] * len(graph)

def dfs(graph, node, visited):
    print(node)
    visited[node] = True

    for neighbor in graph[node]:
        if visited[neighbor] == False:
            dfs(graph, neighbor, visited)

dfs(graph, 1, visited)
```



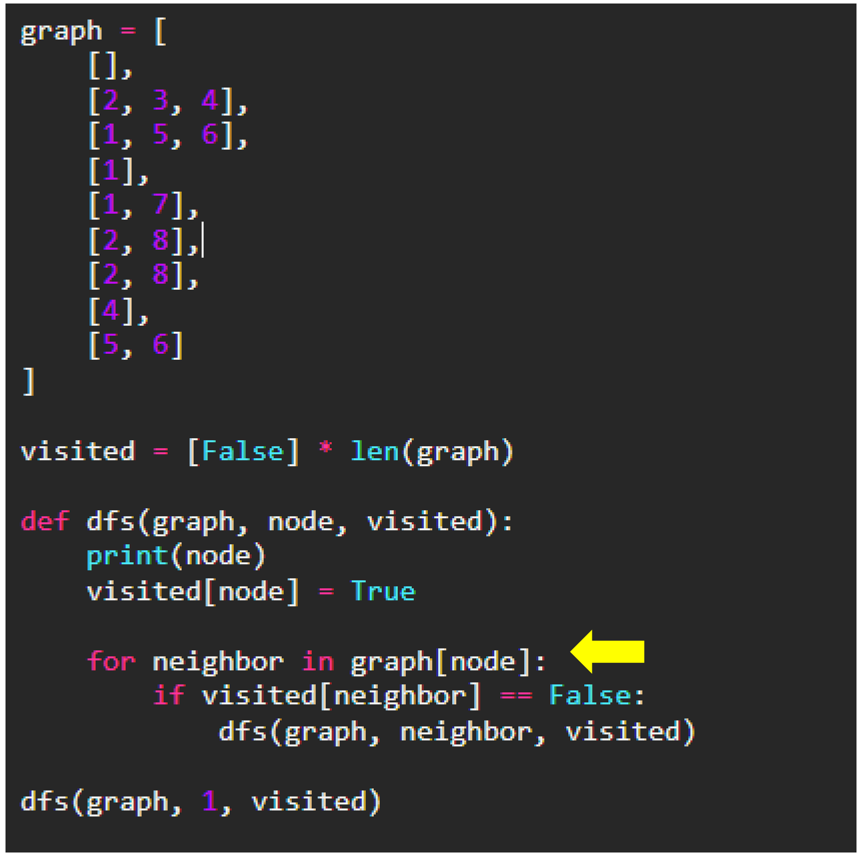
neighbor = 3
neighbor = 4
neighbor = 6



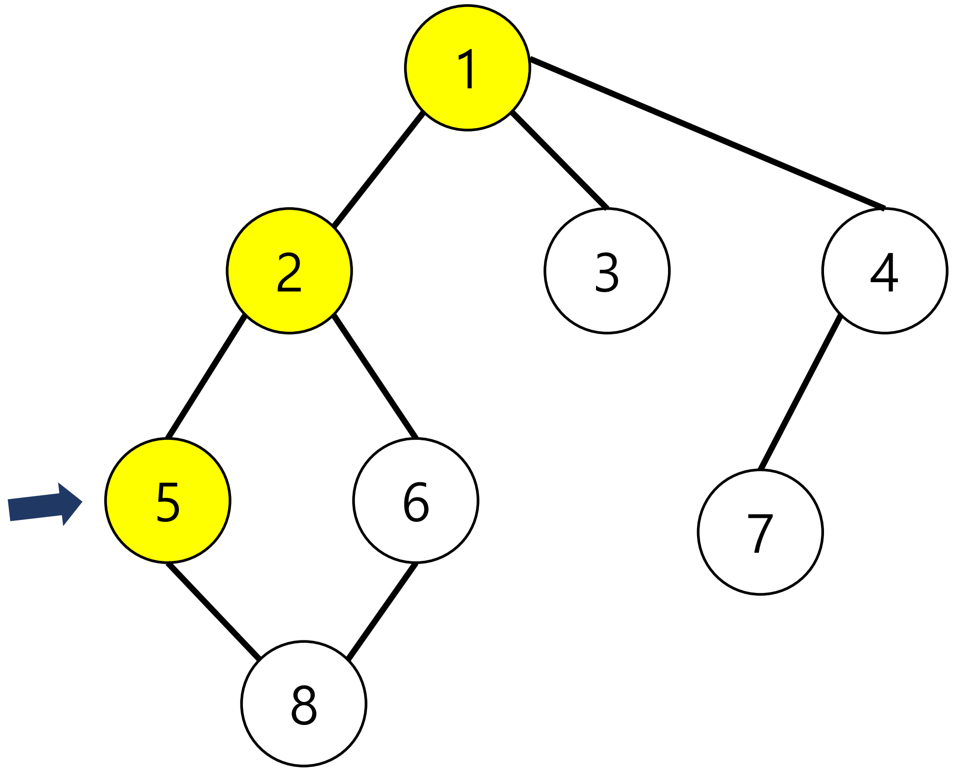
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

visited = [False] * len(graph)

def dfs(graph, node, visited):
    print(node)
    visited[node] = True

    for neighbor in graph[node]:
        if visited[neighbor] == False:
            dfs(graph, neighbor, visited)

dfs(graph, 1, visited)
```



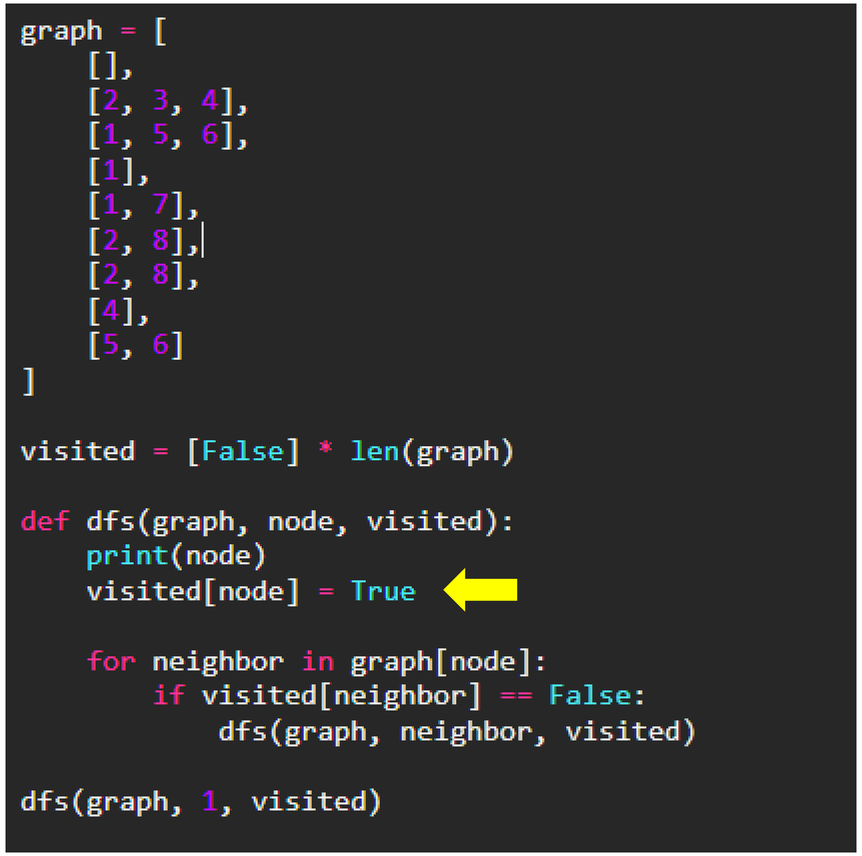
neighbor = 3
neighbor = 4
neighbor = 6



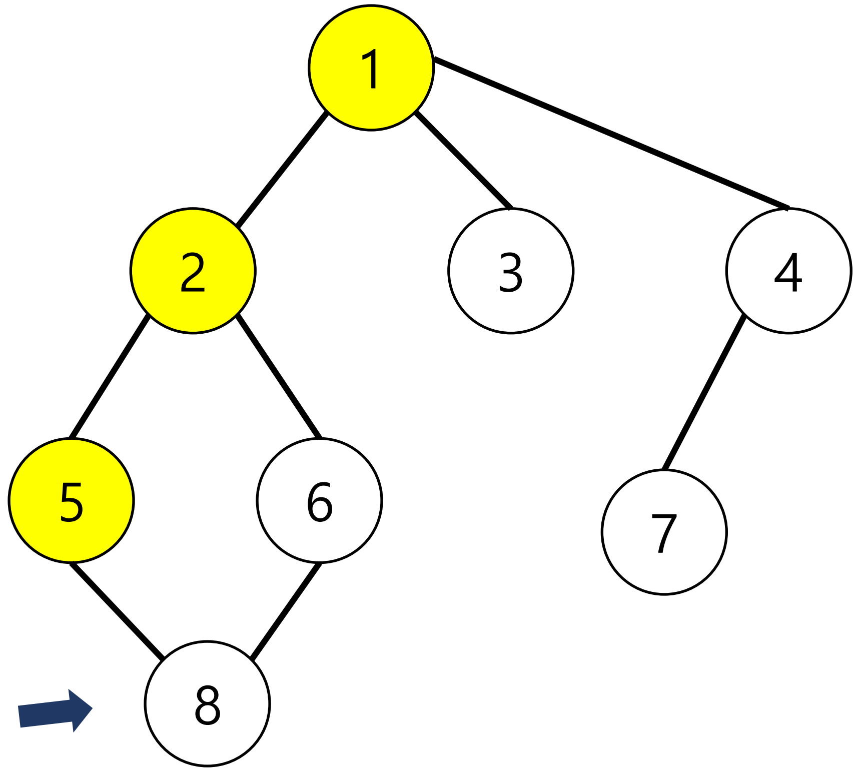
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

visited = [False] * len(graph)

def dfs(graph, node, visited):
    print(node)
    visited[node] = True

    for neighbor in graph[node]:
        if visited[neighbor] == False:
            dfs(graph, neighbor, visited)

dfs(graph, 1, visited)
```



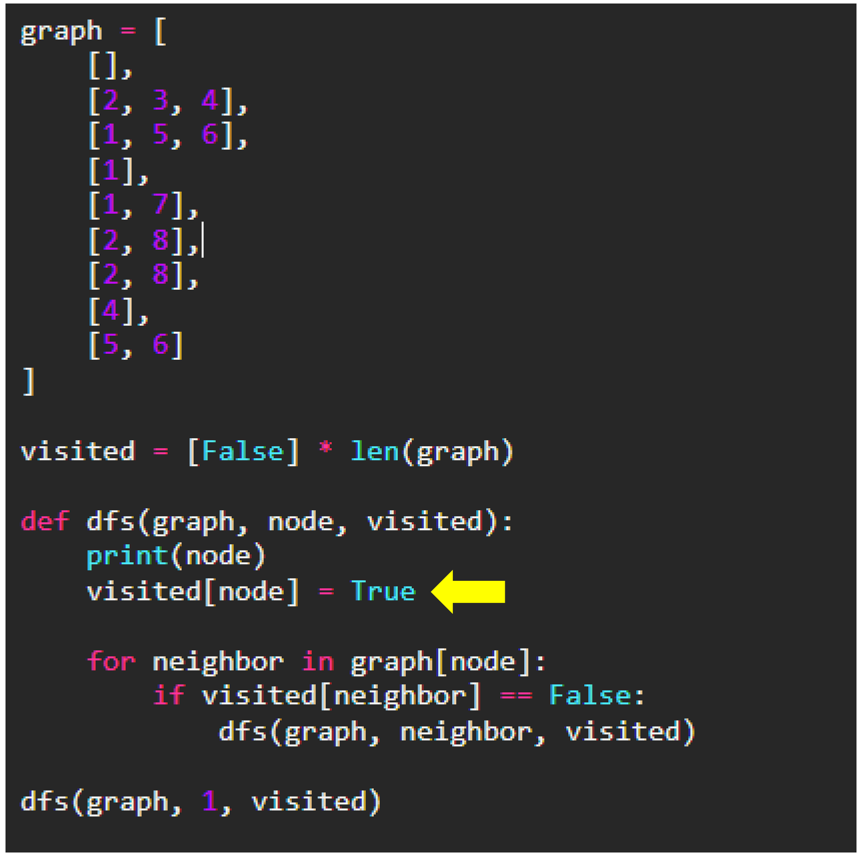
neighbor = 3
neighbor = 4
neighbor = 6



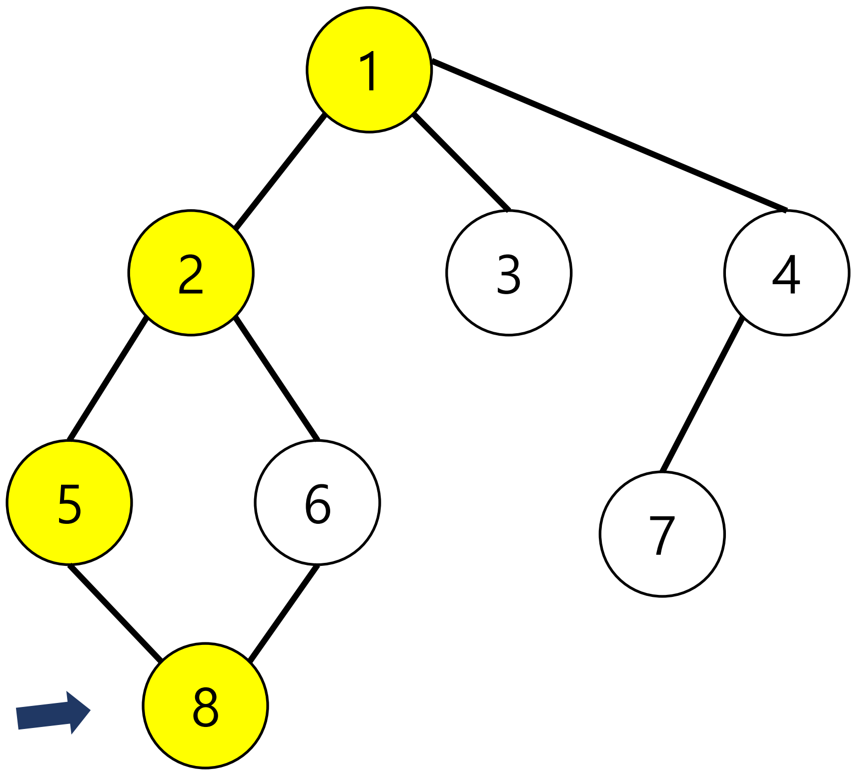
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

visited = [False] * len(graph)

def dfs(graph, node, visited):
    print(node)
    visited[node] = True

    for neighbor in graph[node]:
        if visited[neighbor] == False:
            dfs(graph, neighbor, visited)

dfs(graph, 1, visited)
```



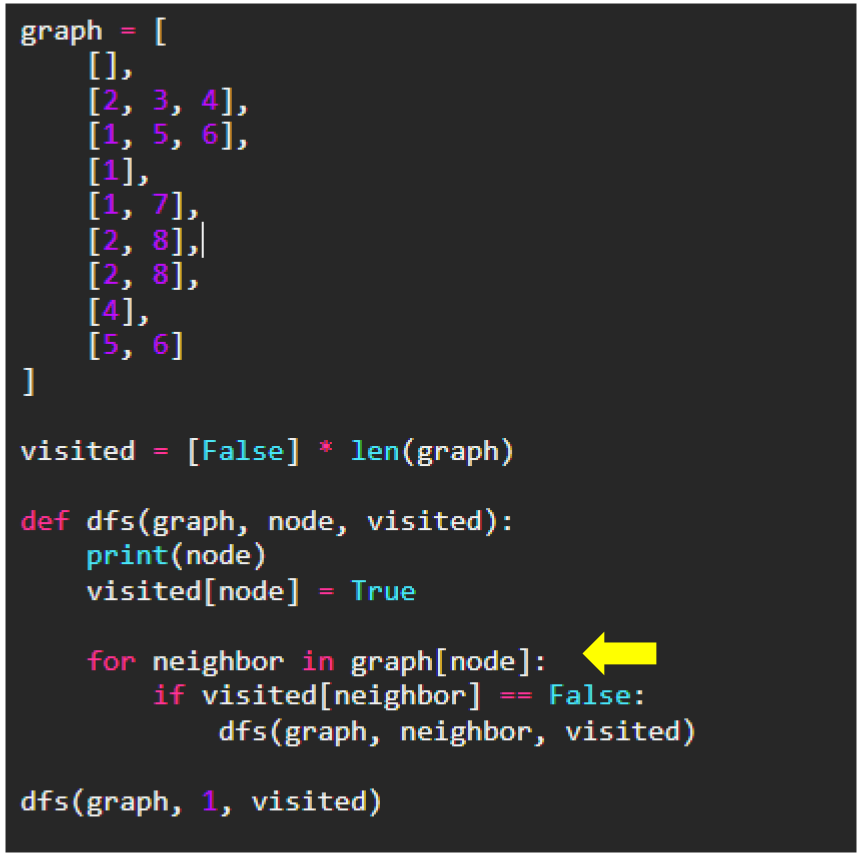
neighbor = 3
neighbor = 4
neighbor = 6



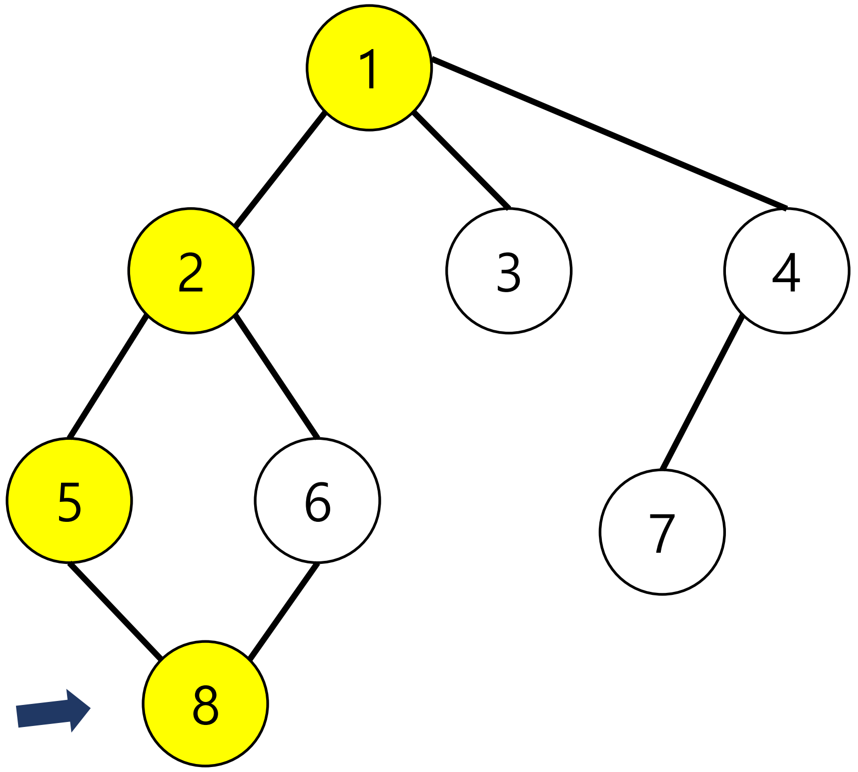
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

visited = [False] * len(graph)

def dfs(graph, node, visited):
    print(node)
    visited[node] = True

    for neighbor in graph[node]:
        if visited[neighbor] == False:
            dfs(graph, neighbor, visited)

dfs(graph, 1, visited)
```



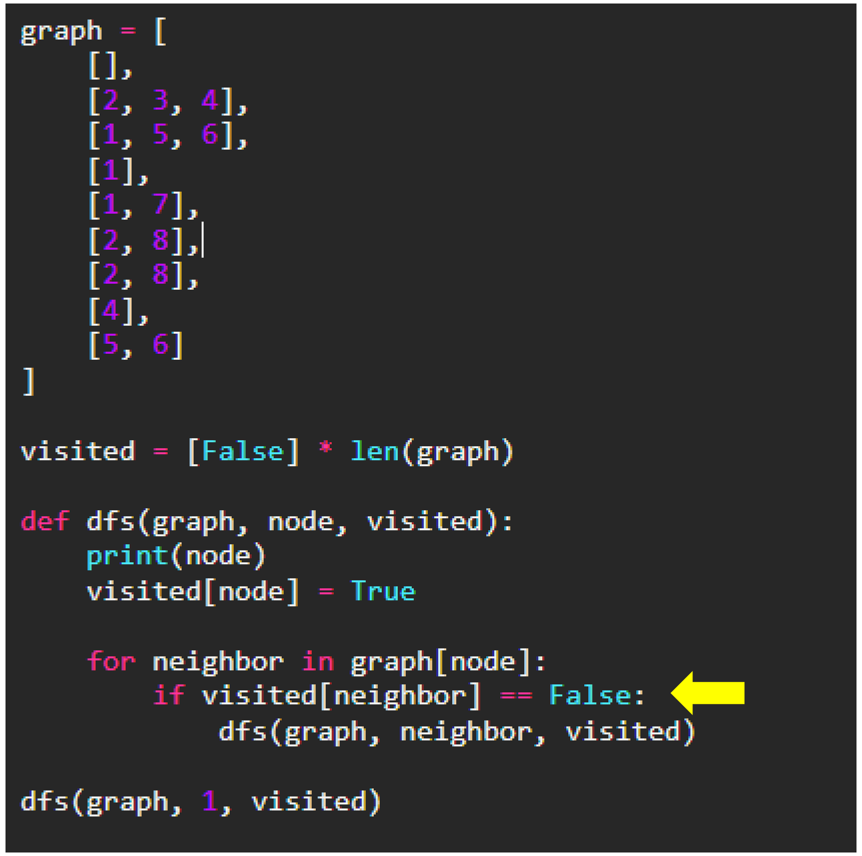
neighbor = 3
neighbor = 4
neighbor = 6



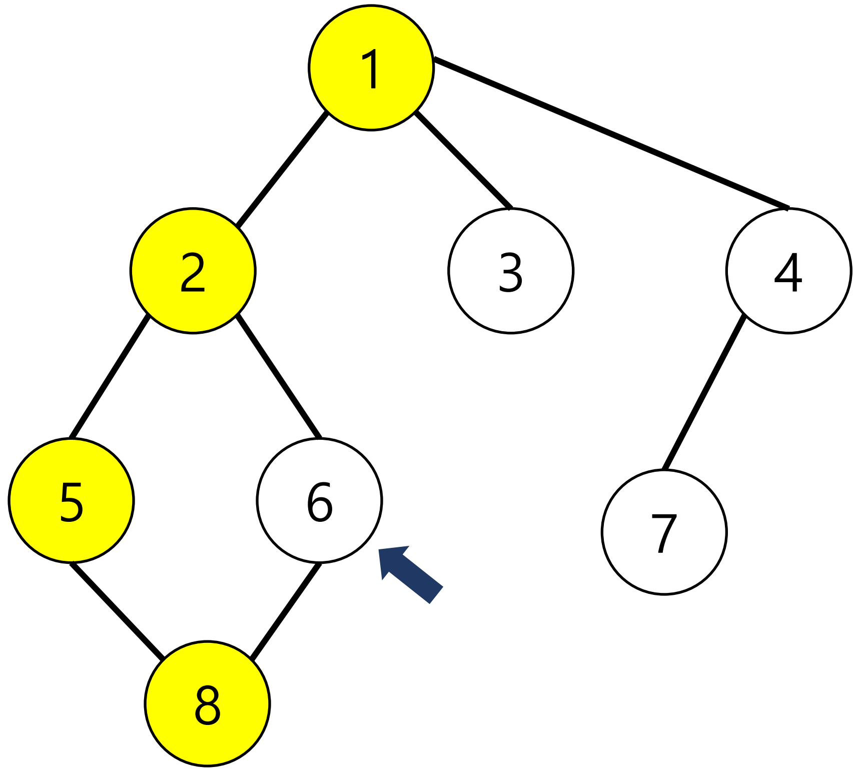
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

visited = [False] * len(graph)

def dfs(graph, node, visited):
    print(node)
    visited[node] = True

    for neighbor in graph[node]:
        if visited[neighbor] == False:
            dfs(graph, neighbor, visited)

dfs(graph, 1, visited)
```



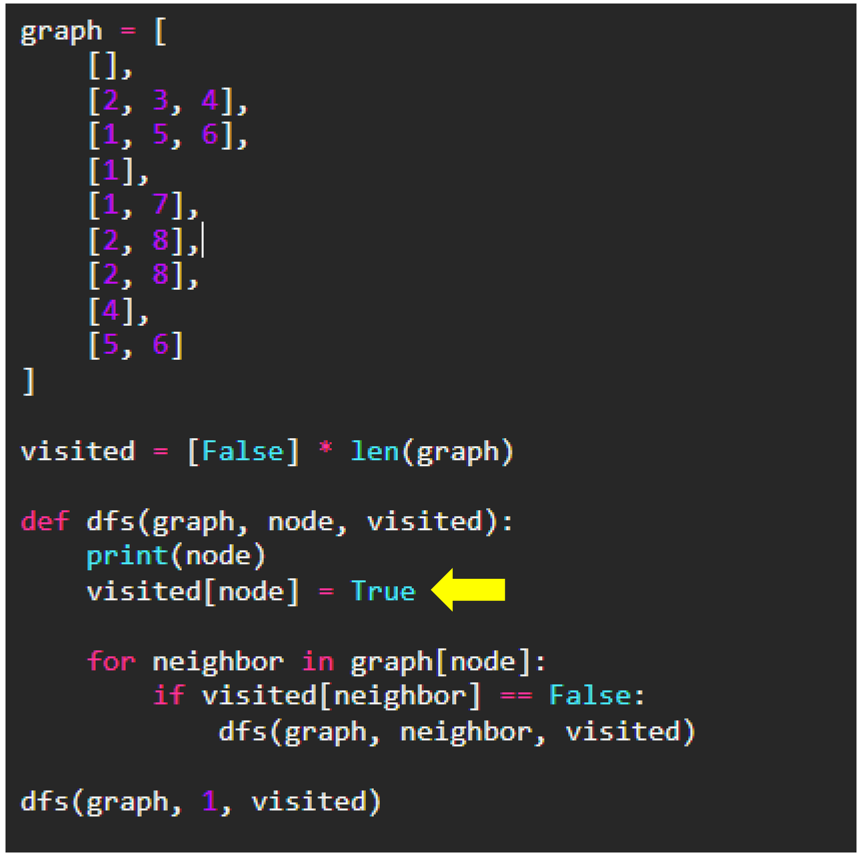
neighbor = 3
neighbor = 4
neighbor = 6



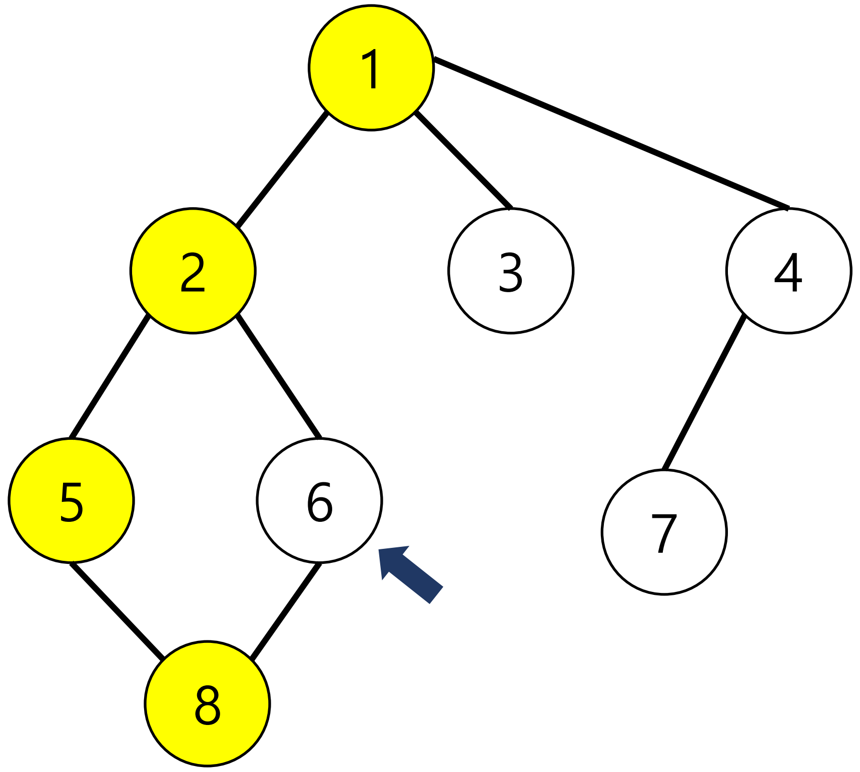
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

visited = [False] * len(graph)

def dfs(graph, node, visited):
    print(node)
    visited[node] = True

    for neighbor in graph[node]:
        if visited[neighbor] == False:
            dfs(graph, neighbor, visited)

dfs(graph, 1, visited)
```



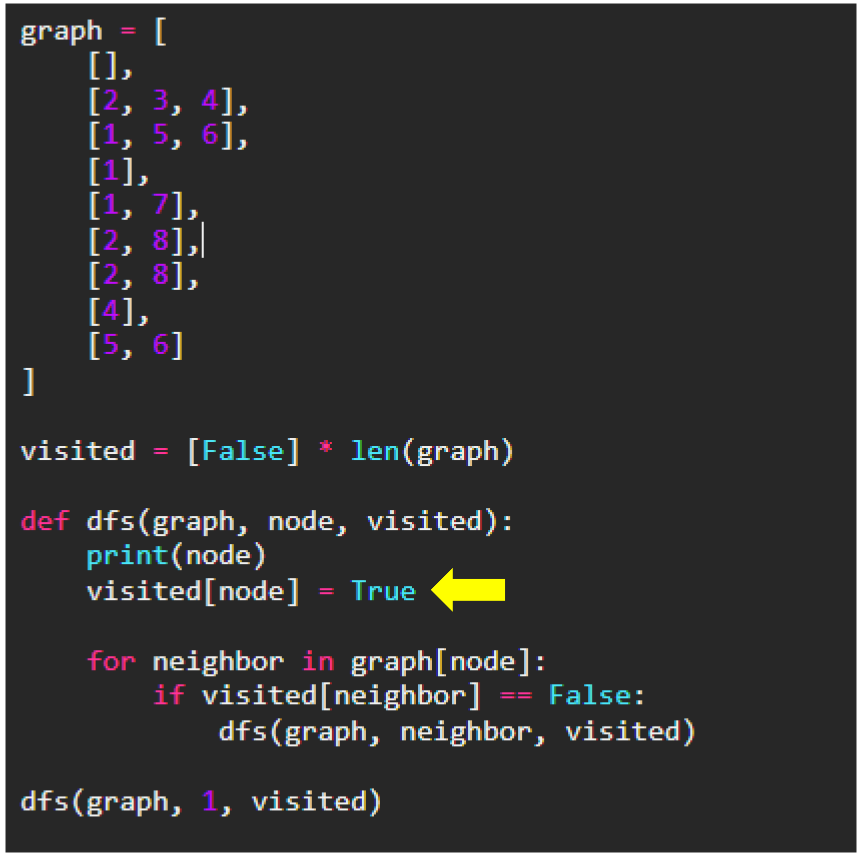
neighbor = 3
neighbor = 4
neighbor = 6



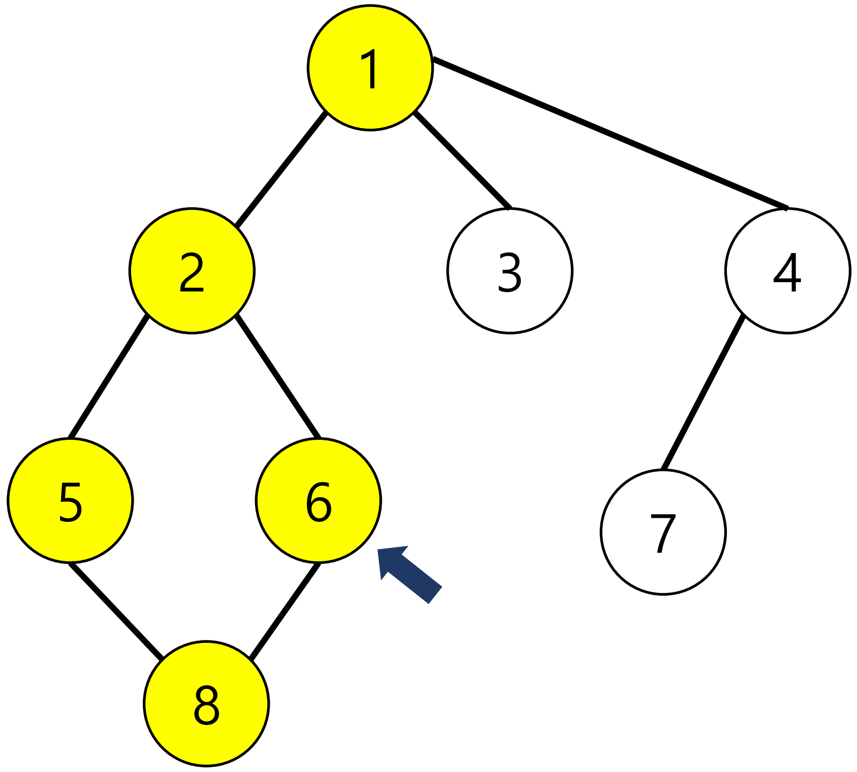
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

visited = [False] * len(graph)

def dfs(graph, node, visited):
    print(node)
    visited[node] = True

    for neighbor in graph[node]:
        if visited[neighbor] == False:
            dfs(graph, neighbor, visited)

dfs(graph, 1, visited)
```



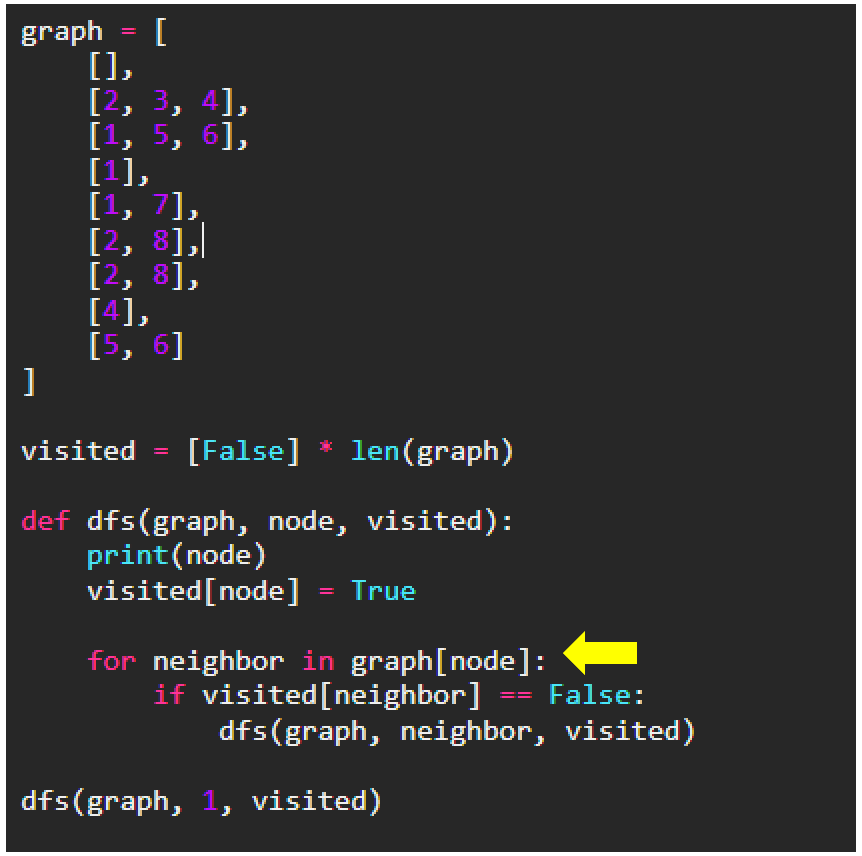
neighbor = 3
neighbor = 4
neighbor = 6



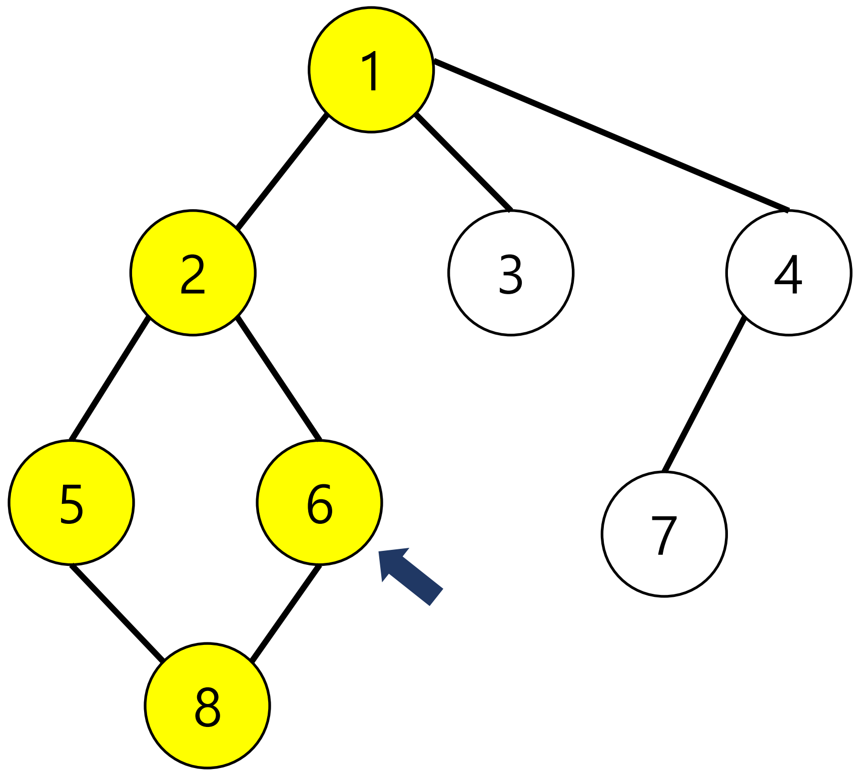
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

visited = [False] * len(graph)

def dfs(graph, node, visited):
    print(node)
    visited[node] = True

    for neighbor in graph[node]:
        if visited[neighbor] == False:
            dfs(graph, neighbor, visited)

dfs(graph, 1, visited)
```



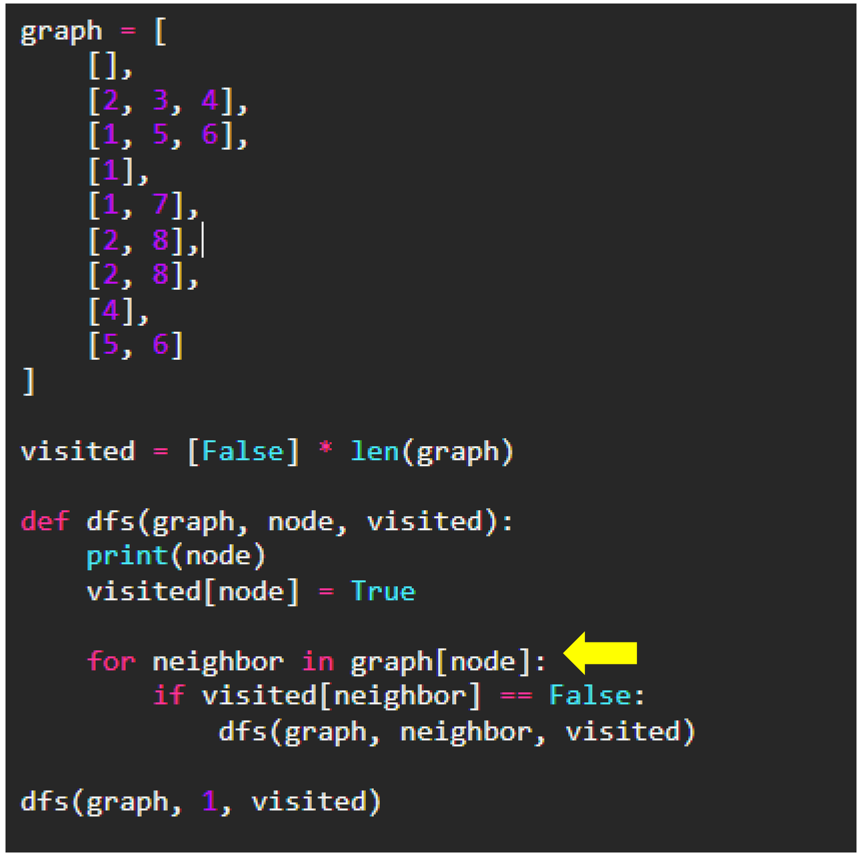
neighbor = 3
neighbor = 4
neighbor = 6



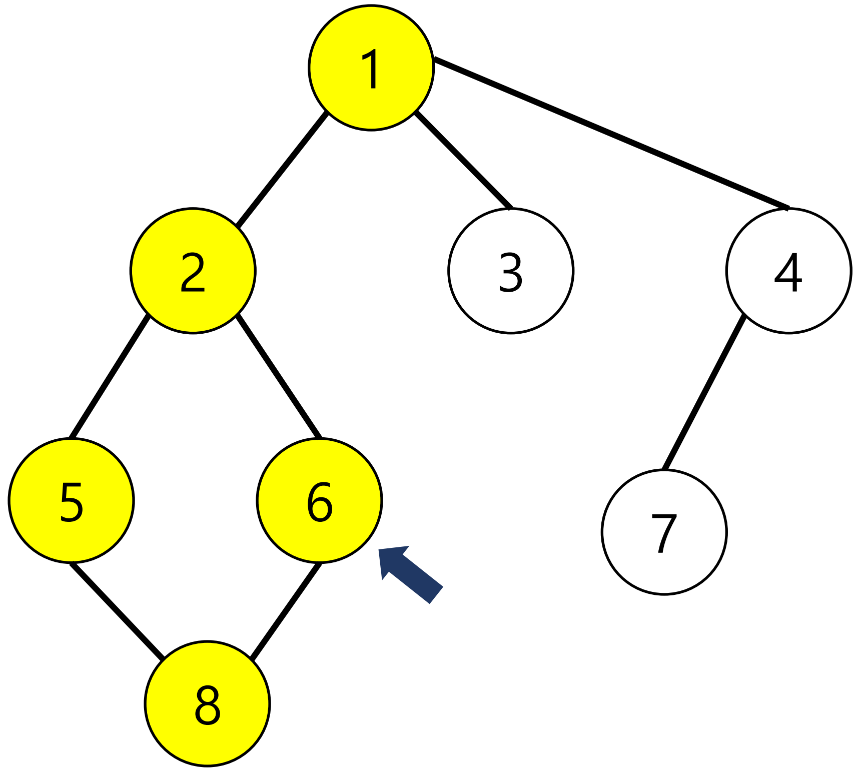
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

visited = [False] * len(graph)

def dfs(graph, node, visited):
    print(node)
    visited[node] = True

    for neighbor in graph[node]:
        if visited[neighbor] == False:
            dfs(graph, neighbor, visited)

dfs(graph, 1, visited)
```



neighbor = 3
neighbor = 4
neighbor = 6

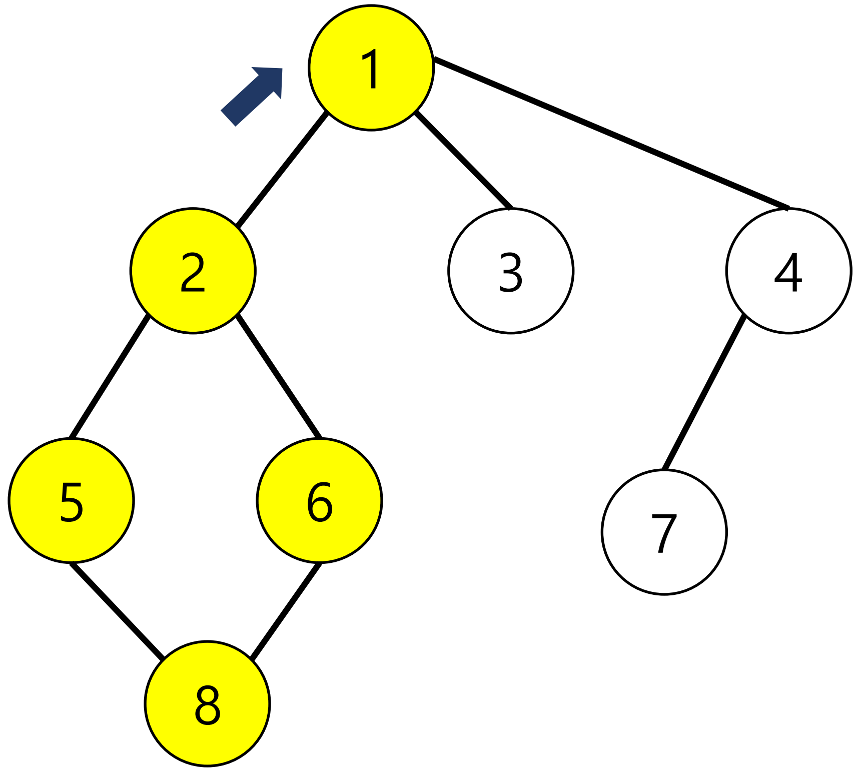


```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

visited = [False] * len(graph)

def dfs(graph, node, visited):
    print(node)
    visited[node] = True

    for neighbor in graph[node]:
        if visited[neighbor] == False:
            dfs(graph, neighbor, visited)

dfs(graph, 1, visited)
```

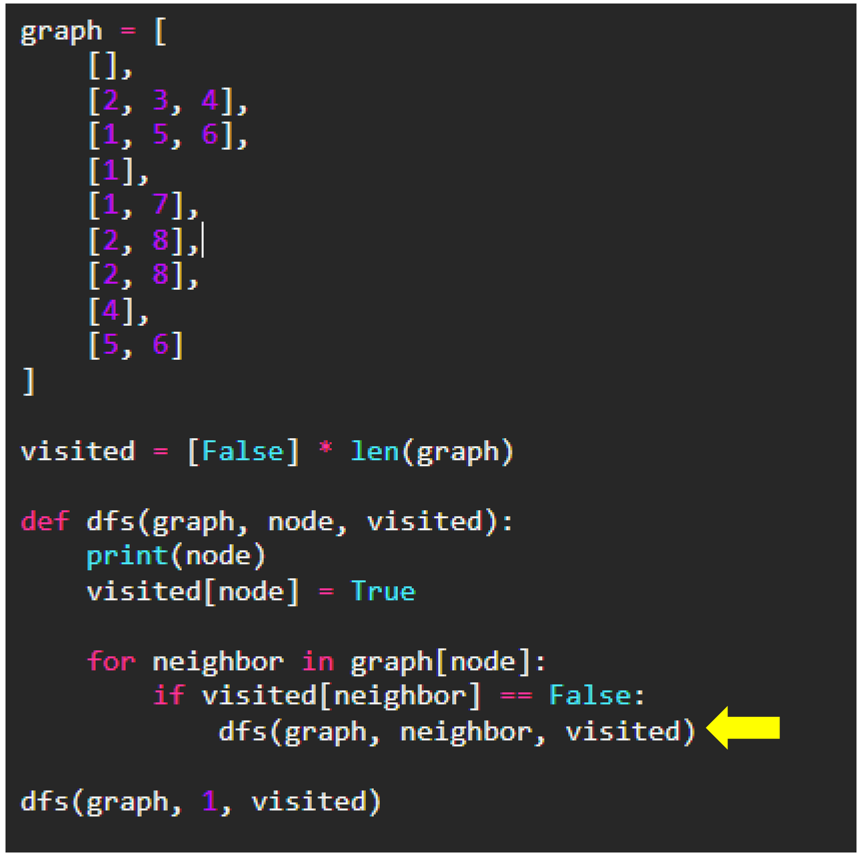


neighbor = 3
neighbor = 4

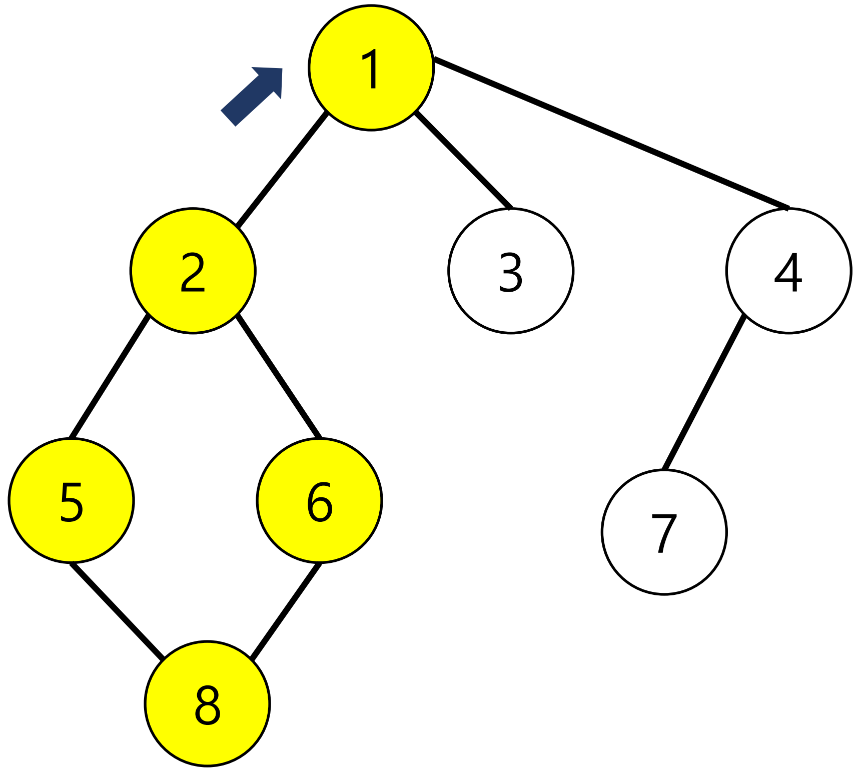
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

visited = [False] * len(graph)

def dfs(graph, node, visited):
    print(node)
    visited[node] = True

    for neighbor in graph[node]:
        if visited[neighbor] == False:
            dfs(graph, neighbor, visited)

dfs(graph, 1, visited)
```



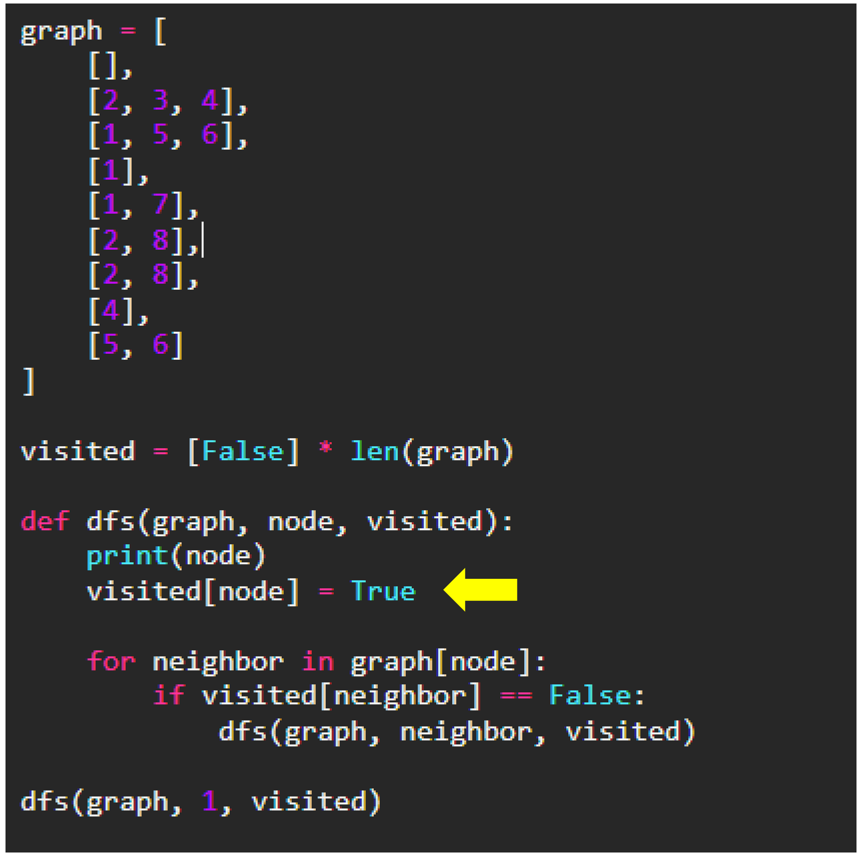
neighbor = 3
neighbor = 4



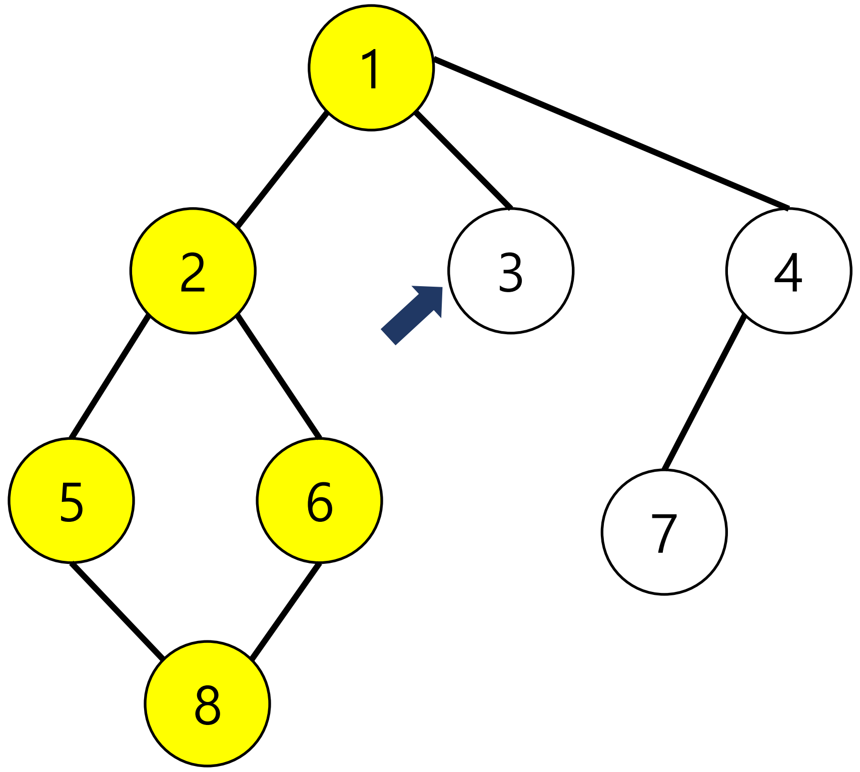
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

visited = [False] * len(graph)

def dfs(graph, node, visited):
    print(node)
    visited[node] = True

    for neighbor in graph[node]:
        if visited[neighbor] == False:
            dfs(graph, neighbor, visited)

dfs(graph, 1, visited)
```



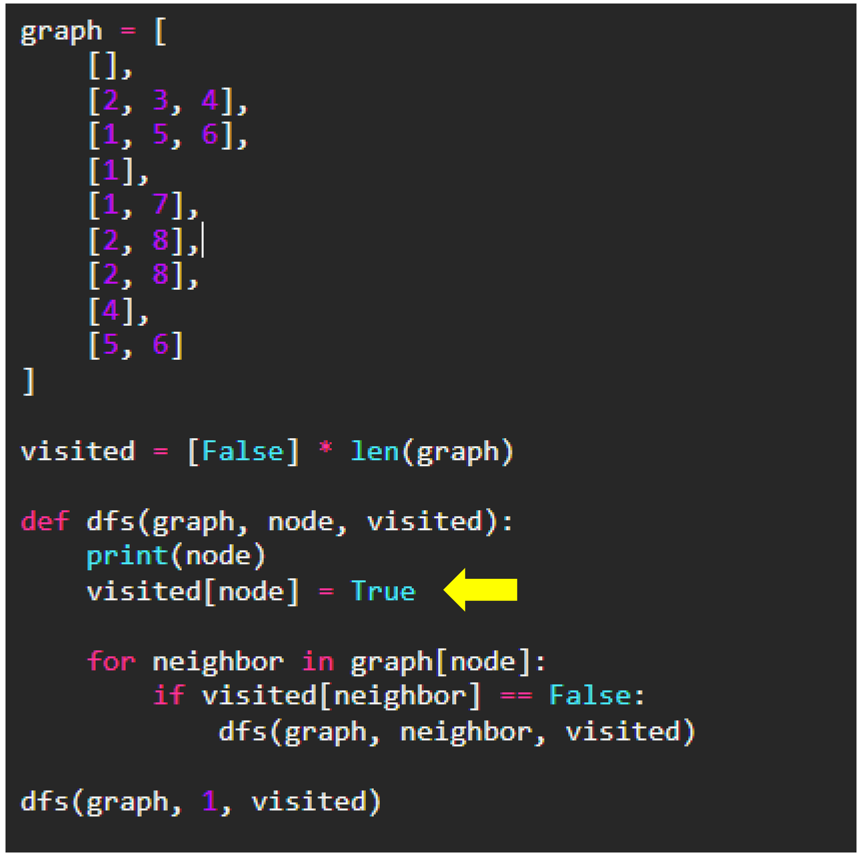
neighbor = 3
neighbor = 4



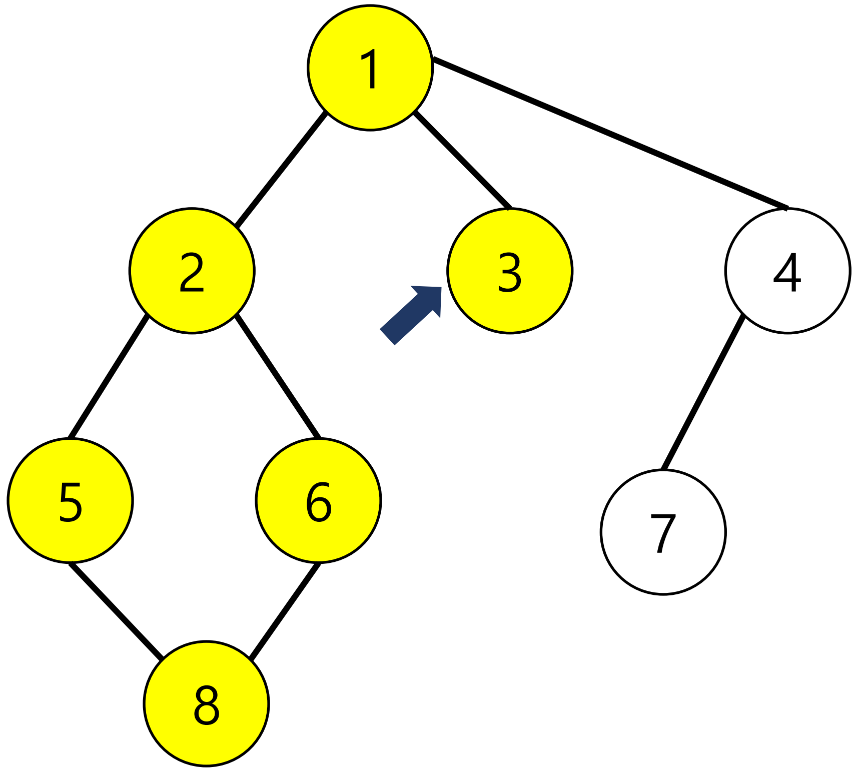
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

visited = [False] * len(graph)

def dfs(graph, node, visited):
    print(node)
    visited[node] = True

    for neighbor in graph[node]:
        if visited[neighbor] == False:
            dfs(graph, neighbor, visited)

dfs(graph, 1, visited)
```



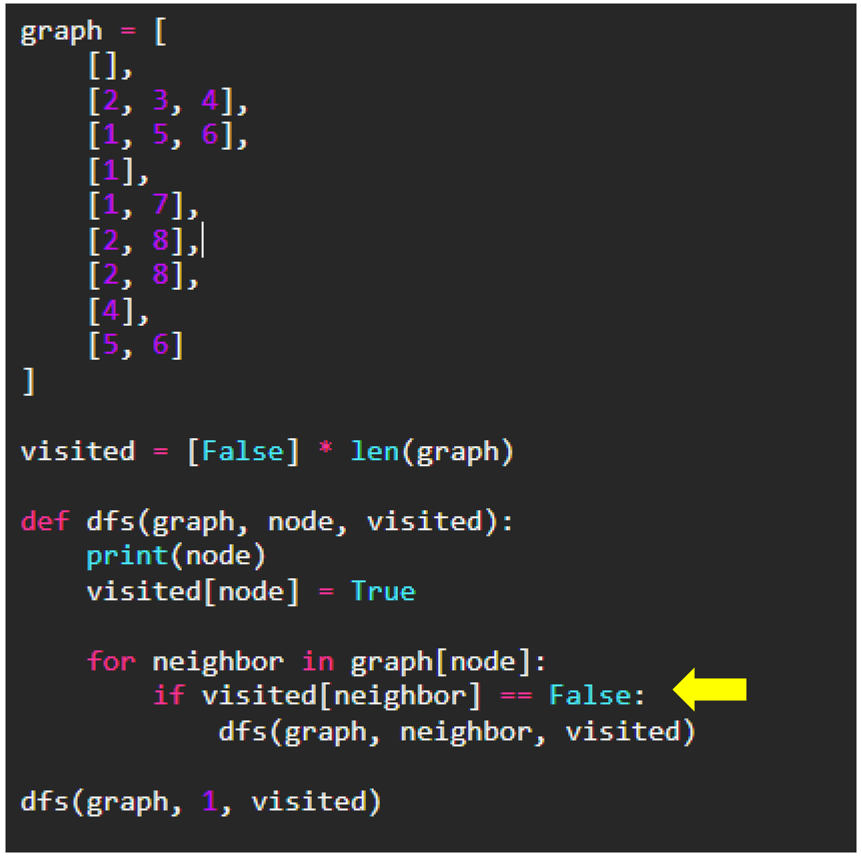
neighbor = 4



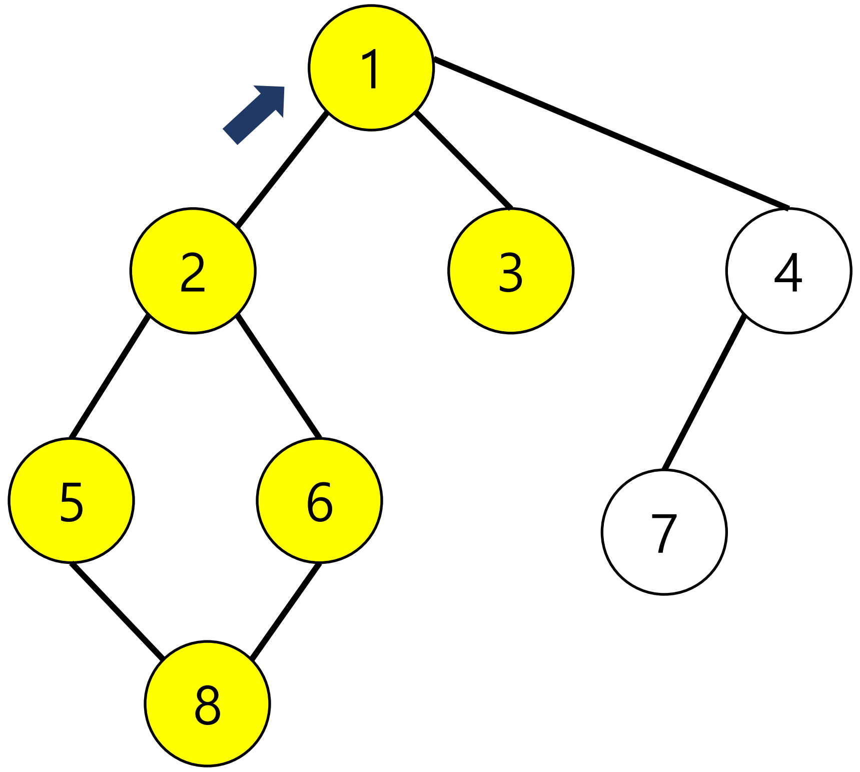
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

visited = [False] * len(graph)

def dfs(graph, node, visited):
    print(node)
    visited[node] = True

    for neighbor in graph[node]:
        if visited[neighbor] == False:
            dfs(graph, neighbor, visited)

dfs(graph, 1, visited)
```



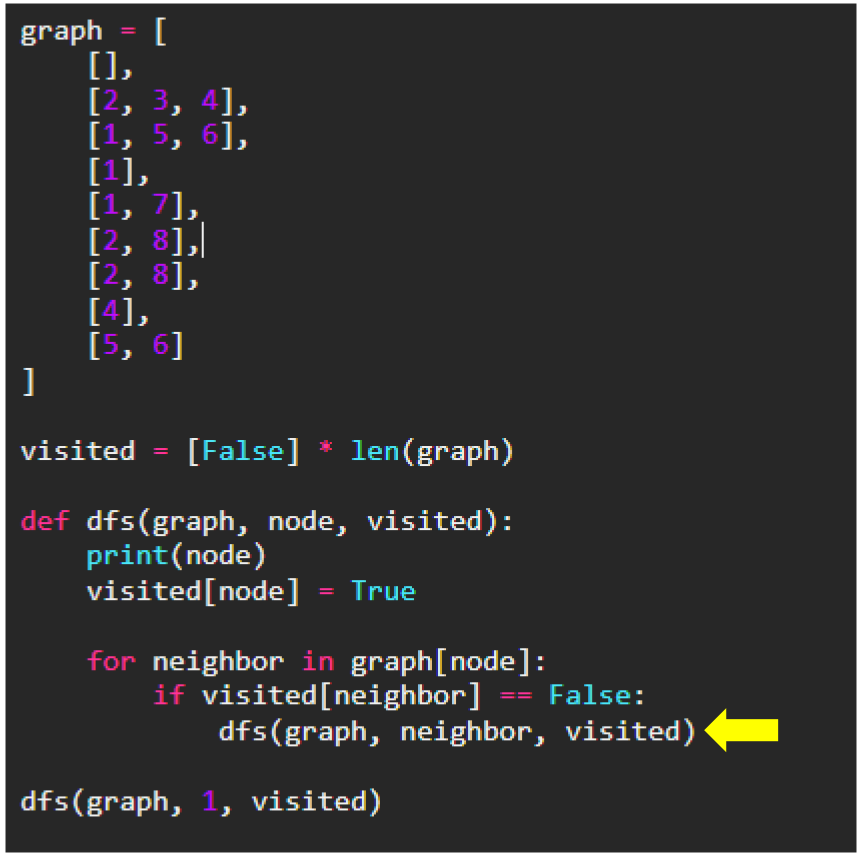
neighbor = 4



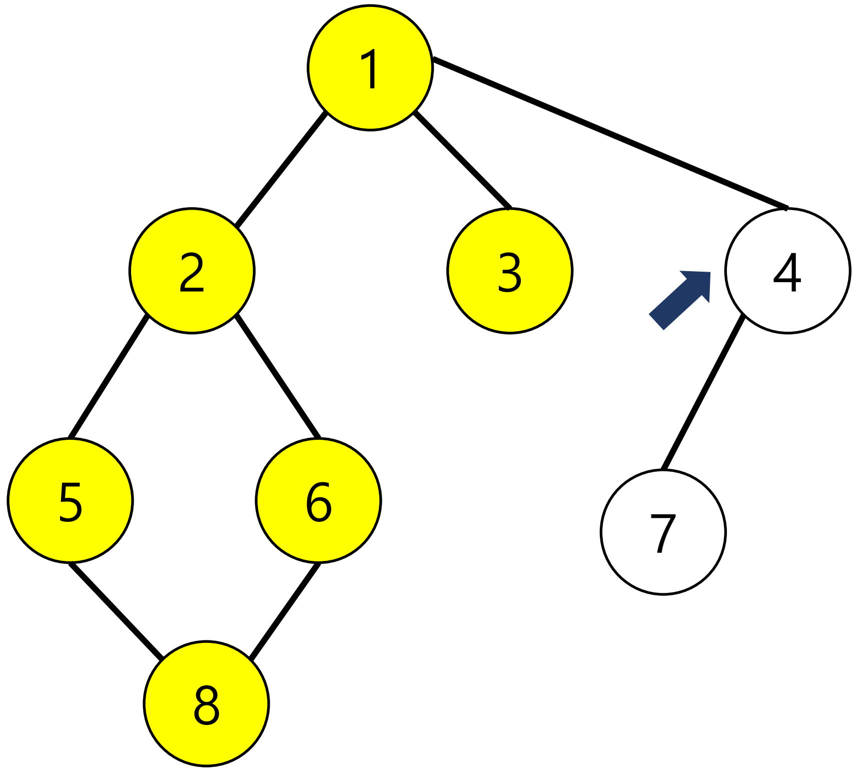
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

visited = [False] * len(graph)

def dfs(graph, node, visited):
    print(node)
    visited[node] = True

    for neighbor in graph[node]:
        if visited[neighbor] == False:
            dfs(graph, neighbor, visited)

dfs(graph, 1, visited)
```

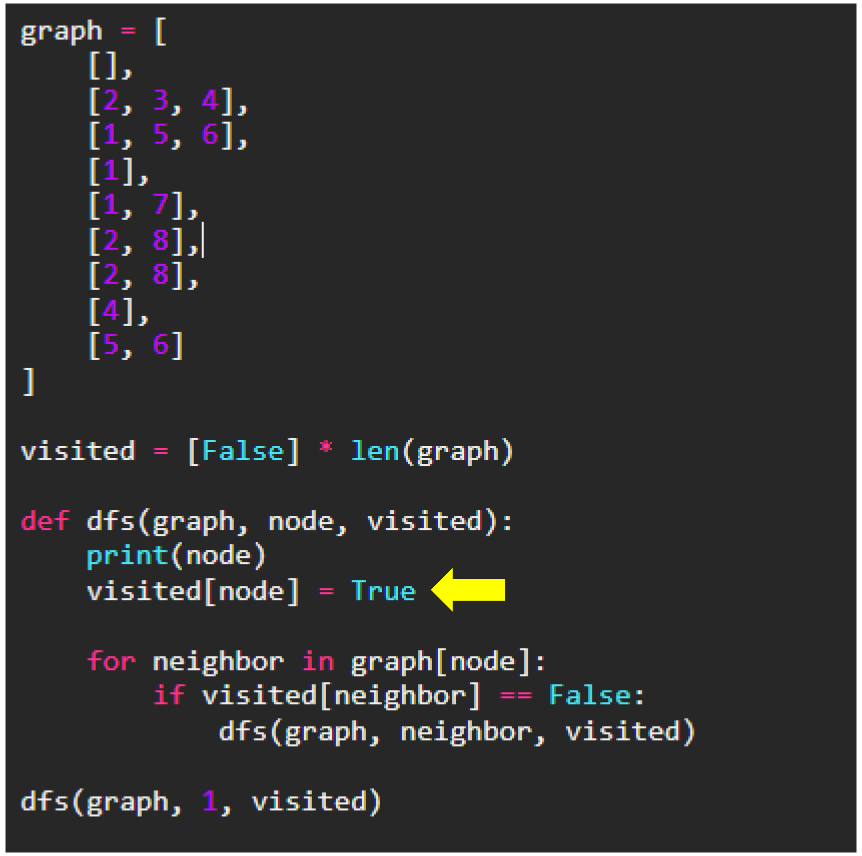


neighbor = 4

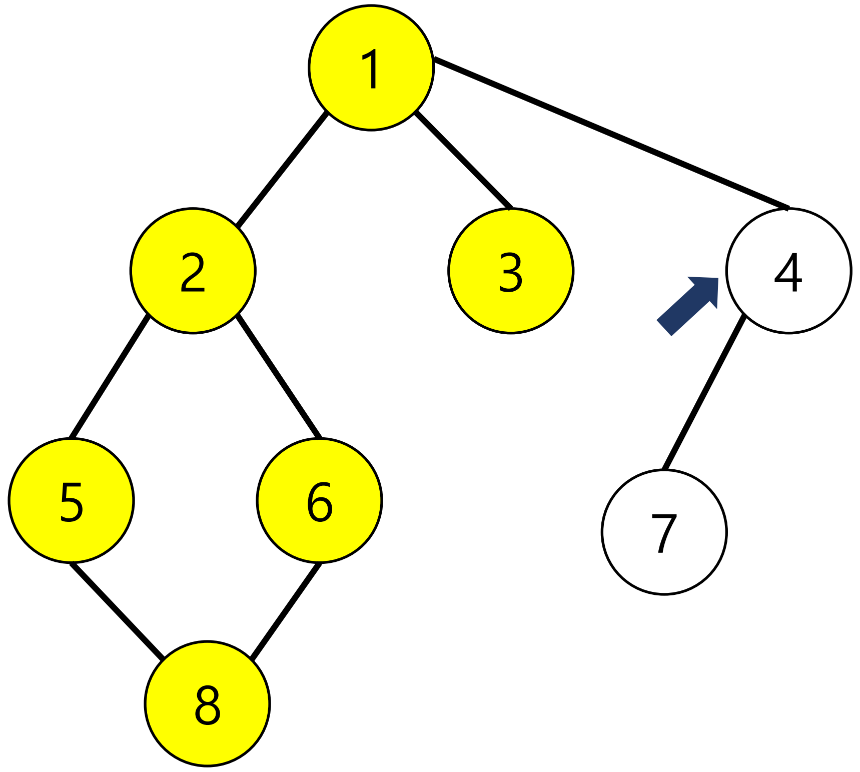


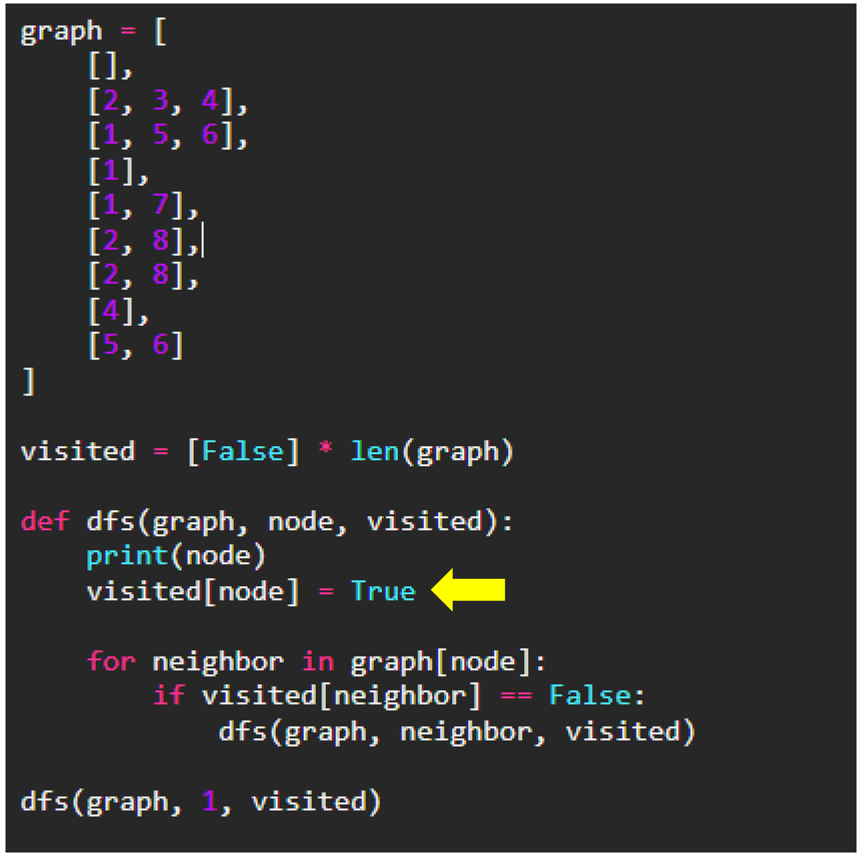
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

visited = [False] * len(graph)

def dfs(graph, node, visited):
    print(node)
    visited[node] = True

    for neighbor in graph[node]:
        if visited[neighbor] == False:
            dfs(graph, neighbor, visited)

dfs(graph, 1, visited)
```



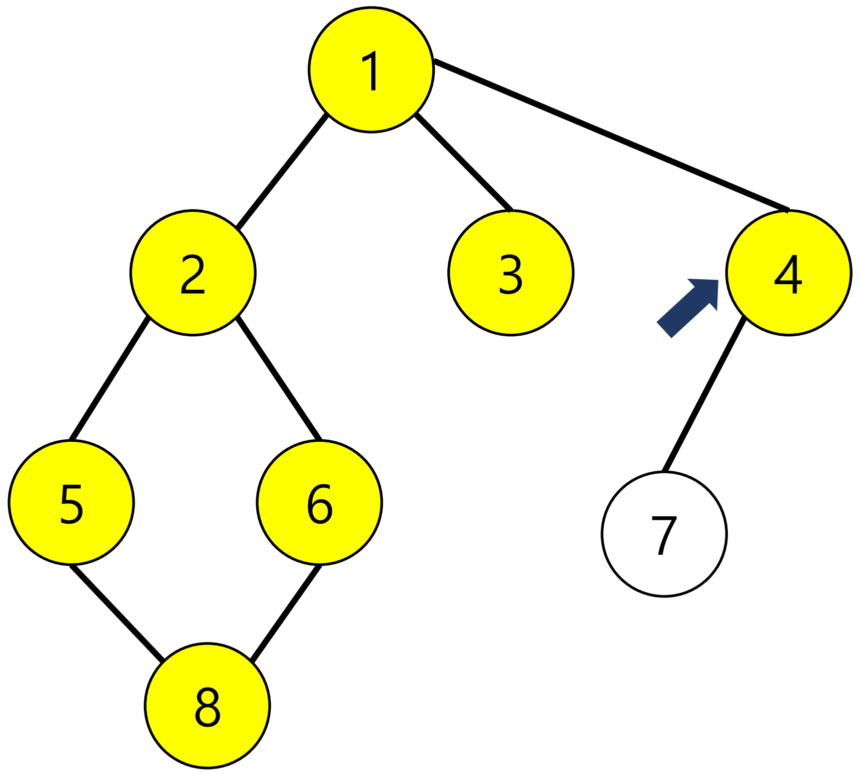


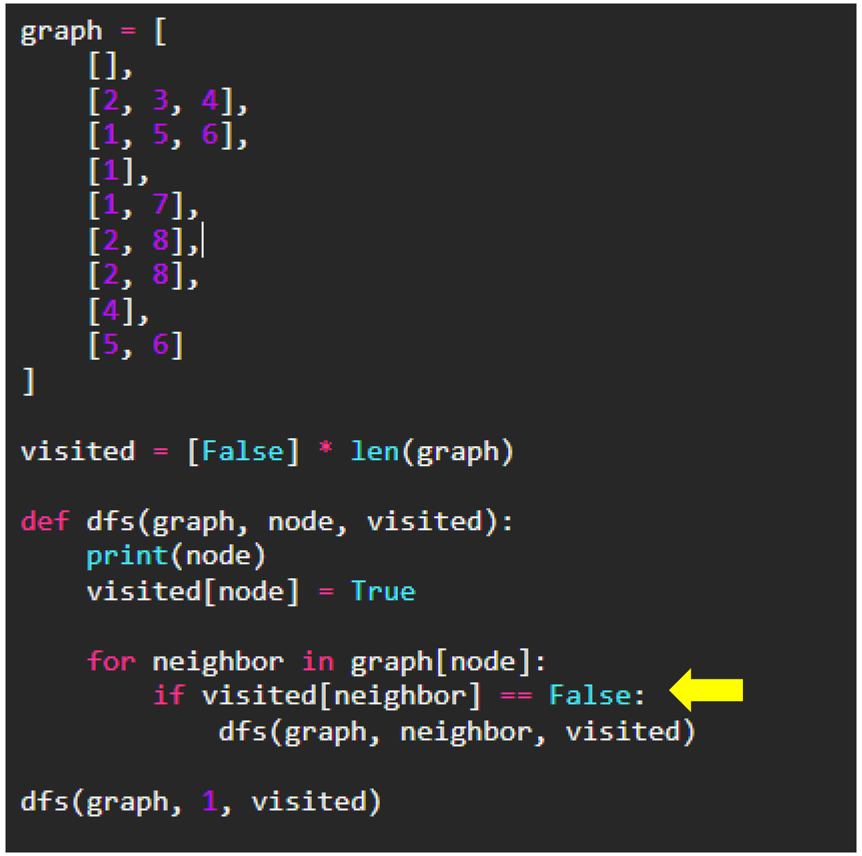
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],|
    [2, 8],
    [4],
    [5, 6]
]

visited = [False] * len(graph)

def dfs(graph, node, visited):
    print(node)
    visited[node] = True

    for neighbor in graph[node]:
        if visited[neighbor] == False:
            dfs(graph, neighbor, visited)

dfs(graph, 1, visited)
```



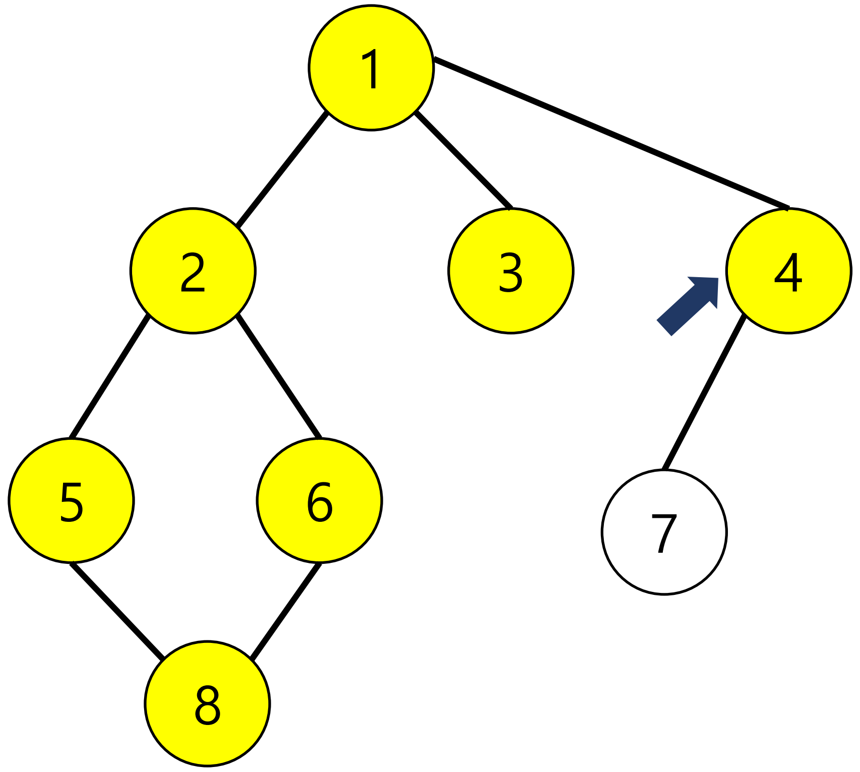


```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

visited = [False] * len(graph)

def dfs(graph, node, visited):
    print(node)
    visited[node] = True

    for neighbor in graph[node]:
        if visited[neighbor] == False:
            dfs(graph, neighbor, visited)

dfs(graph, 1, visited)
```

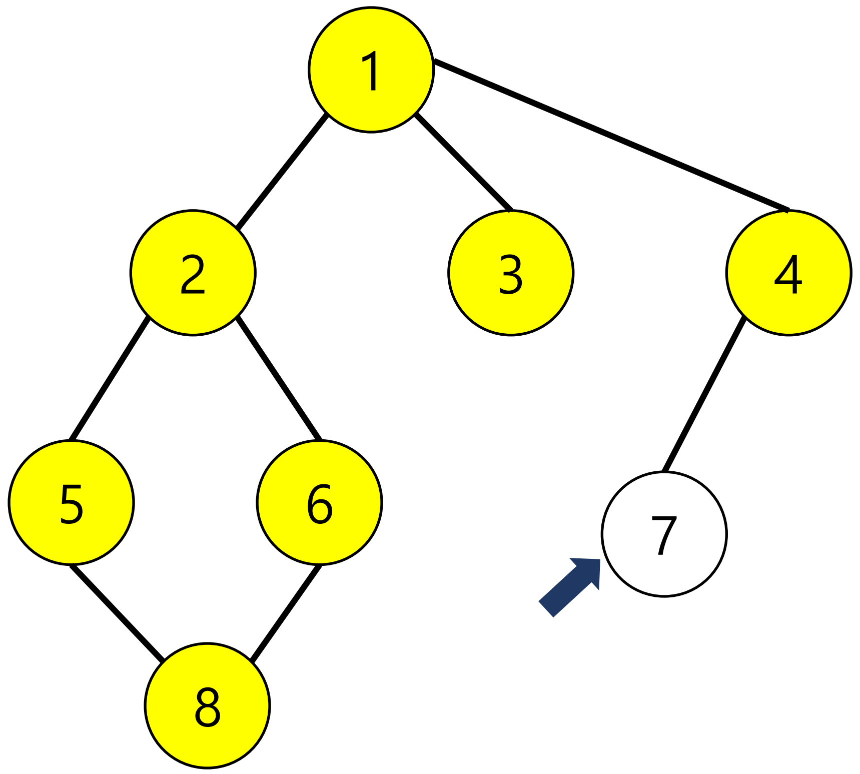


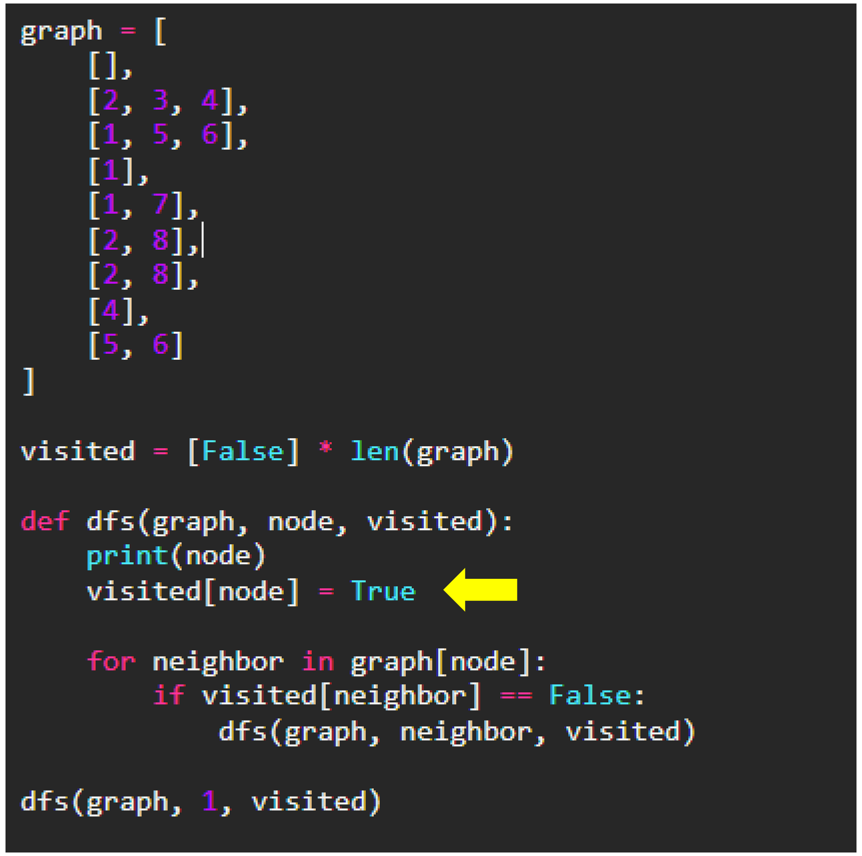


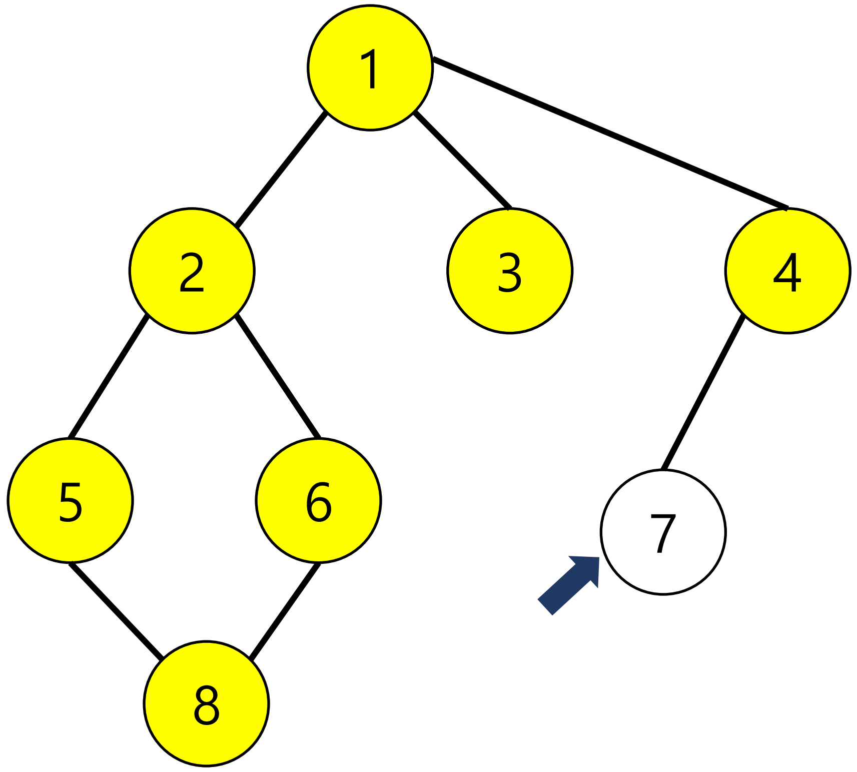
```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

visited = [False] * len(graph)

def dfs(graph, node, visited):
    print(node)
    visited[node] = True

    for neighbor in graph[node]:
        if visited[neighbor] == False:
            dfs(graph, neighbor, visited)

dfs(graph, 1, visited)
```

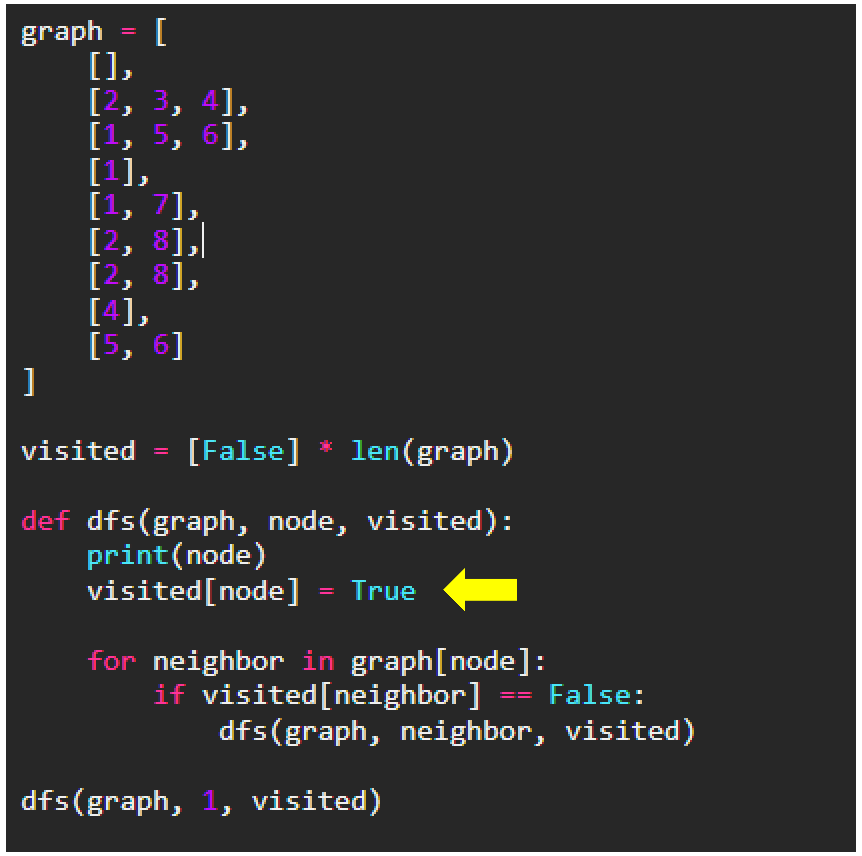


```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

visited = [False] * len(graph)

def dfs(graph, node, visited):
    print(node)
    visited[node] = True

    for neighbor in graph[node]:
        if visited[neighbor] == False:
            dfs(graph, neighbor, visited)

dfs(graph, 1, visited)
```

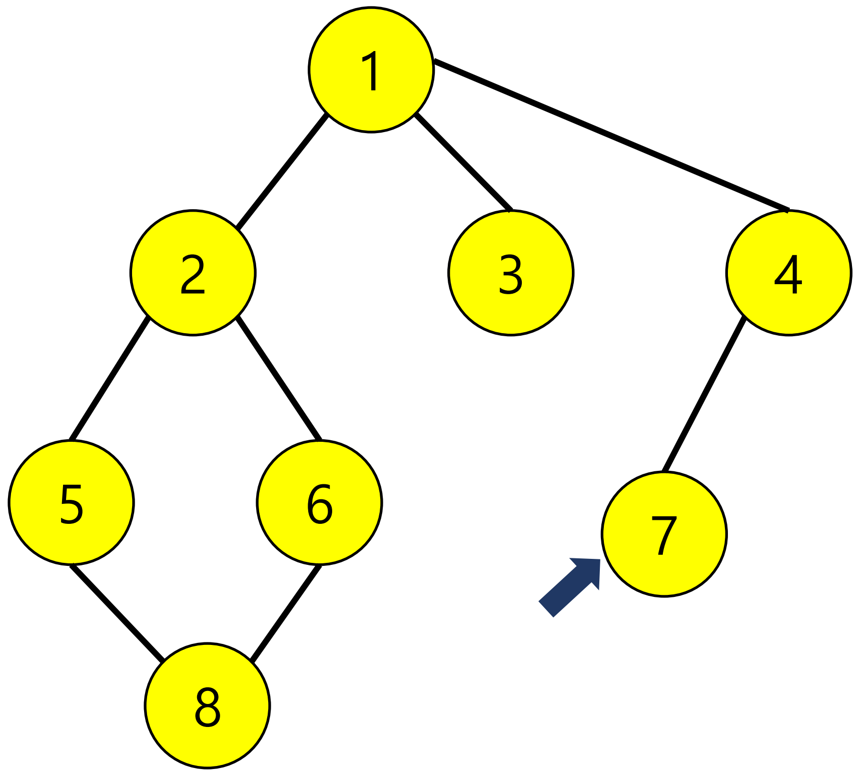



```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

visited = [False] * len(graph)

def dfs(graph, node, visited):
    print(node)
    visited[node] = True

    for neighbor in graph[node]:
        if visited[neighbor] == False:
            dfs(graph, neighbor, visited)

dfs(graph, 1, visited)
```





```
graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

visited = [False] * len(graph)

def dfs(graph, node, visited):
    print(node)
    visited[node] = True

    for neighbor in graph[node]:
        if visited[neighbor] == False:
            dfs(graph, neighbor, visited)

dfs(graph, 1, visited)
```



BFS 너비 우선 탐색

‘큐‘ 자료구조를 사용한 그래프 탐색 알고리즘

현재 노드와 가까운(인접) 노드를 우선적으로 넓게 탐색하는 방식

1. 루트 노드를 큐에 넣고 방문처리한다.
2. 큐를 Deque하고, Deque한 노드의 방문하지 않은 모든 인접 노드를 큐에 넣고 방문 처리한다.
3. 2단계를 더 이상 수행할 수 없을 때 까지 반복한다.

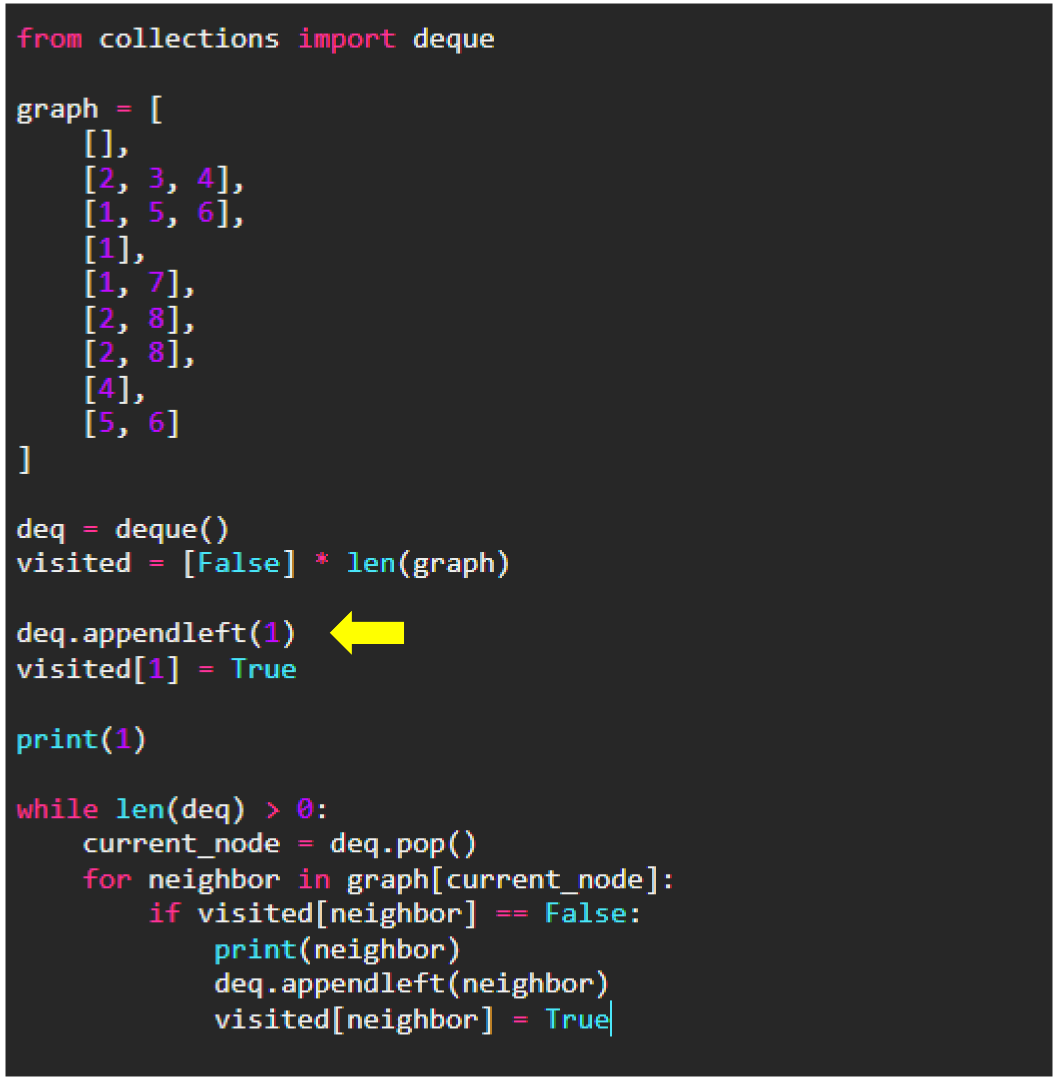
BFS 너비 우선 탐색

장점

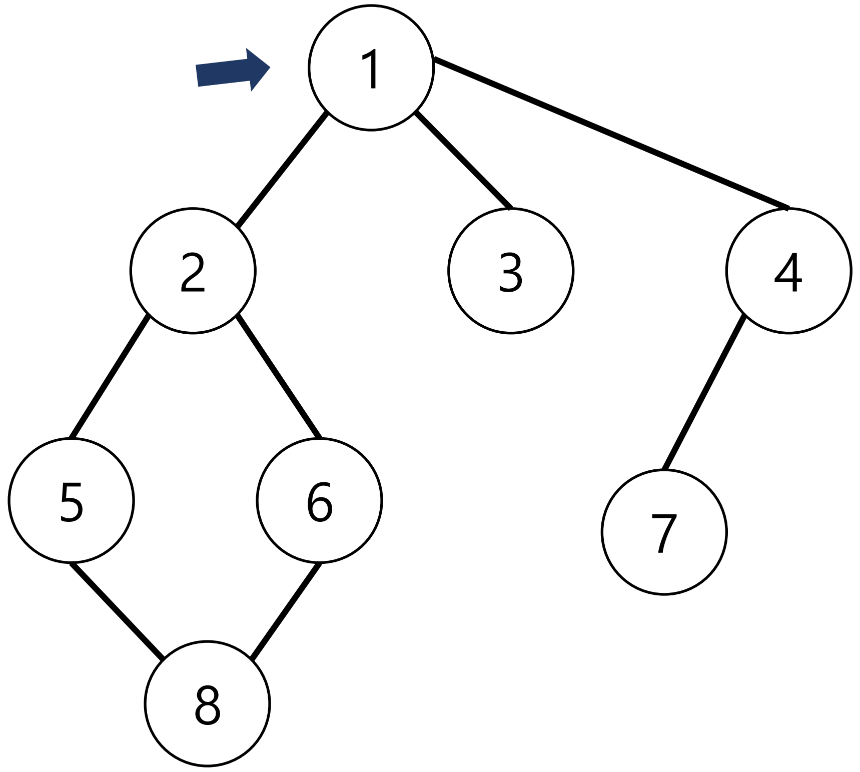
- 노드의 수가 적고 깊이가 얕은 경우 빠르게 동작할 수 있음
- 출발 노드에서 목표 노드까지의 최단 길이 경로를 보장한다.

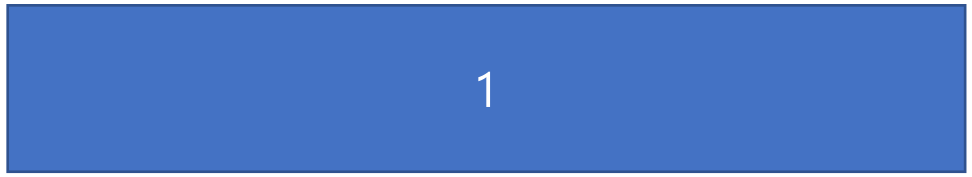
단점

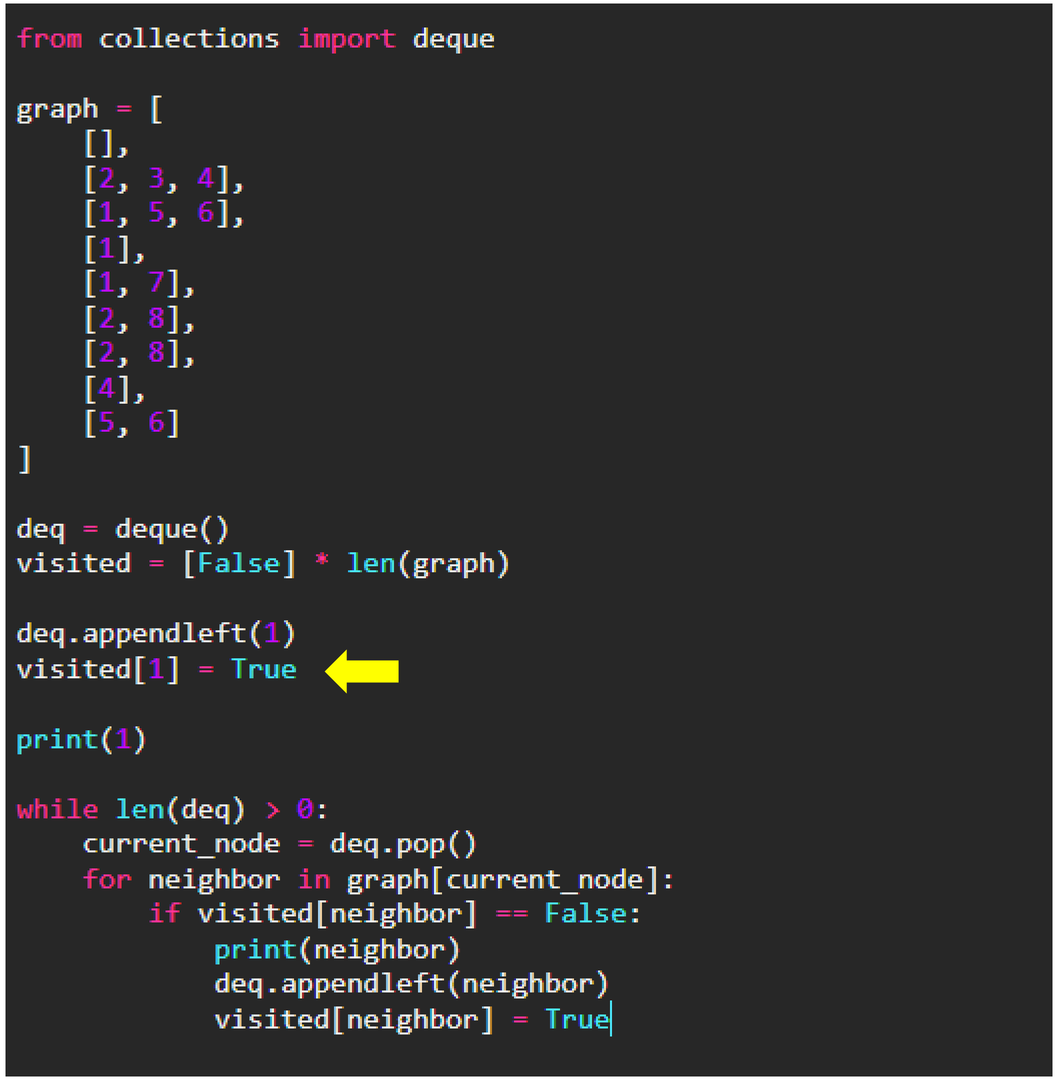
- 최악의 경우 모든 노드에 대한 정보를 저장할 공간을 요구한다.



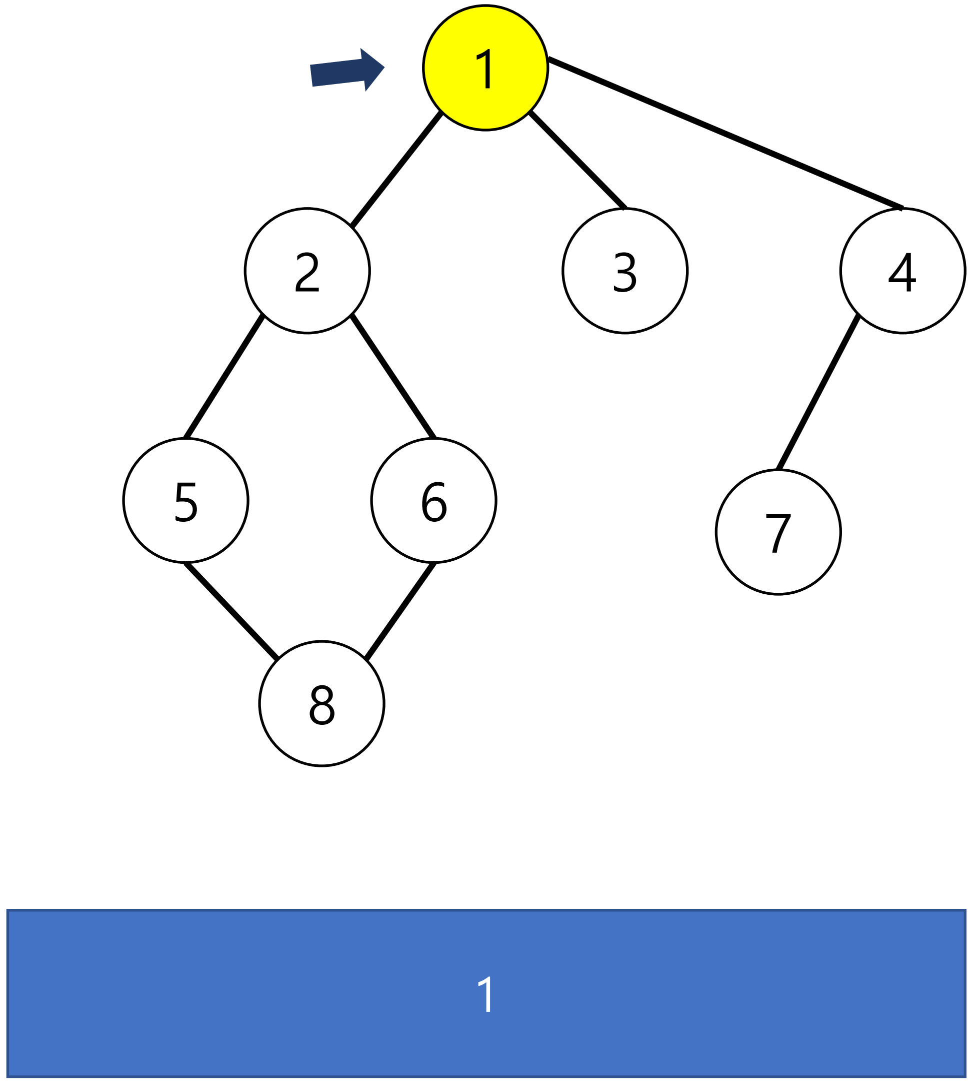
```
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```

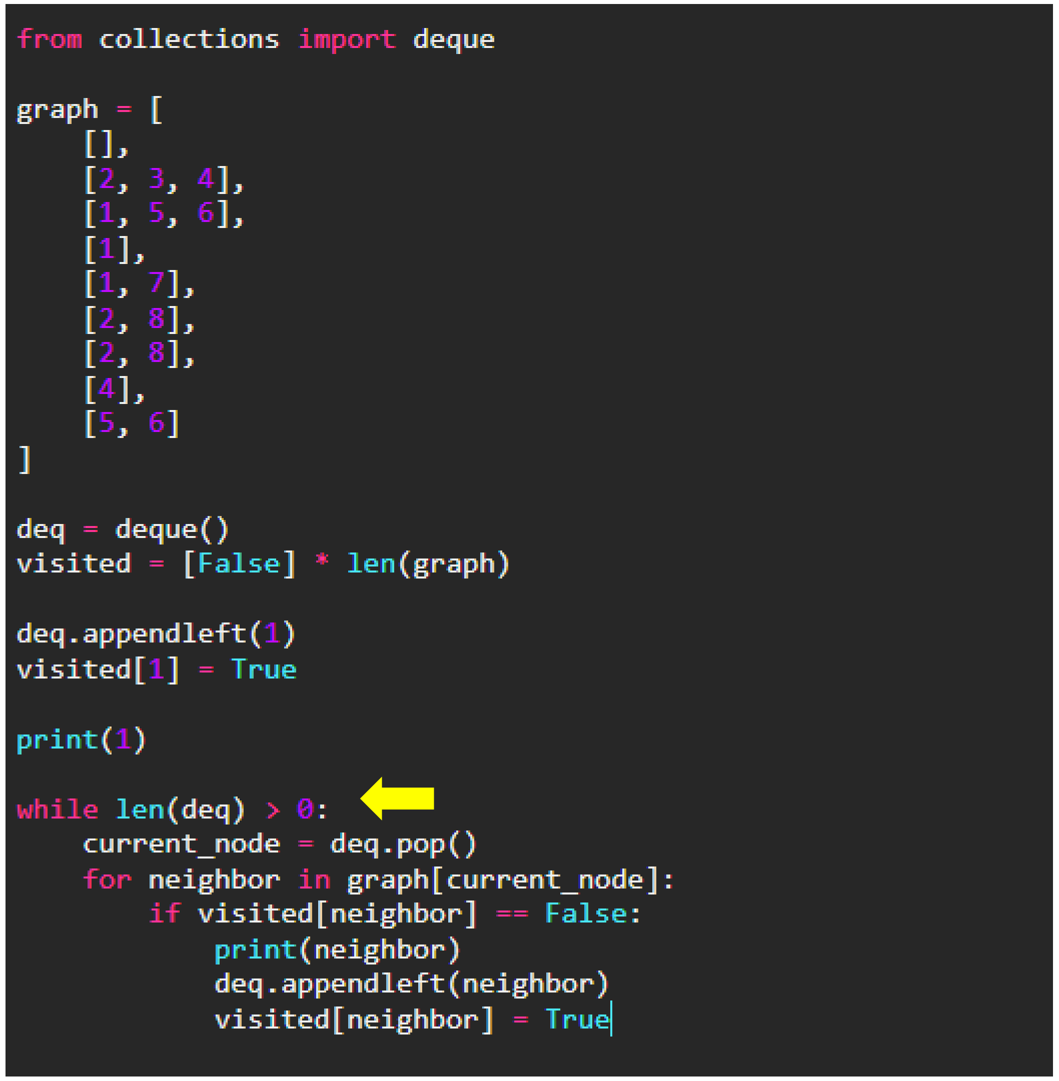




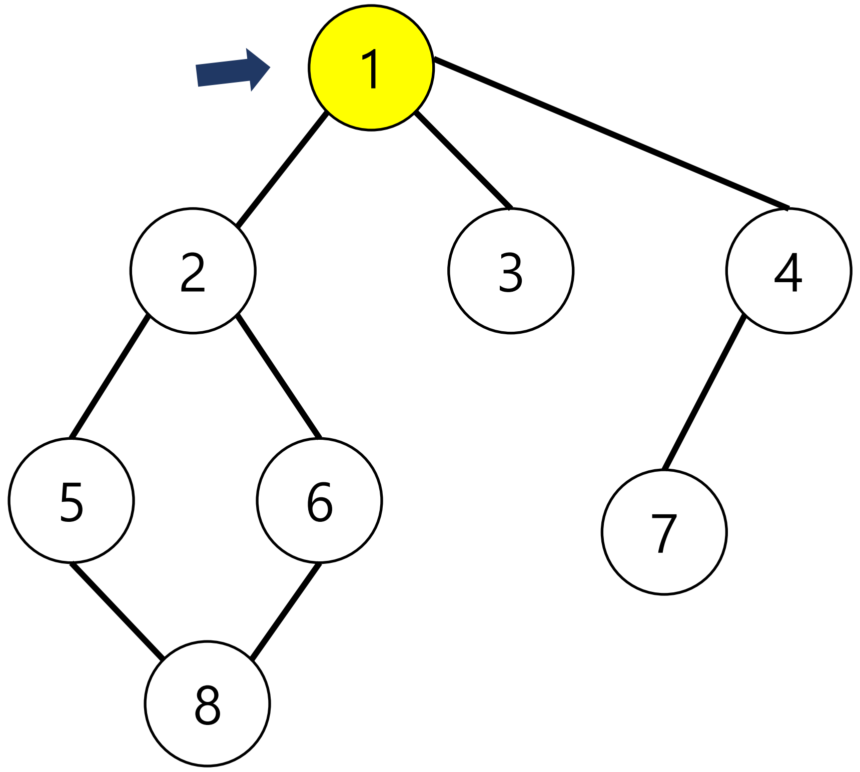


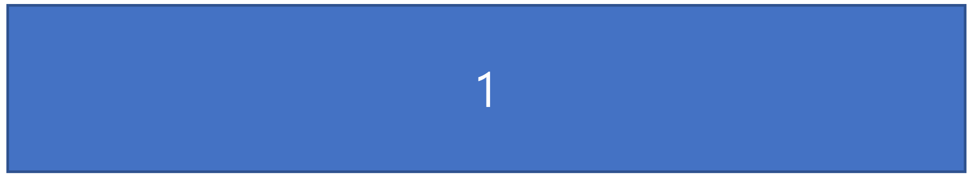
```
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```

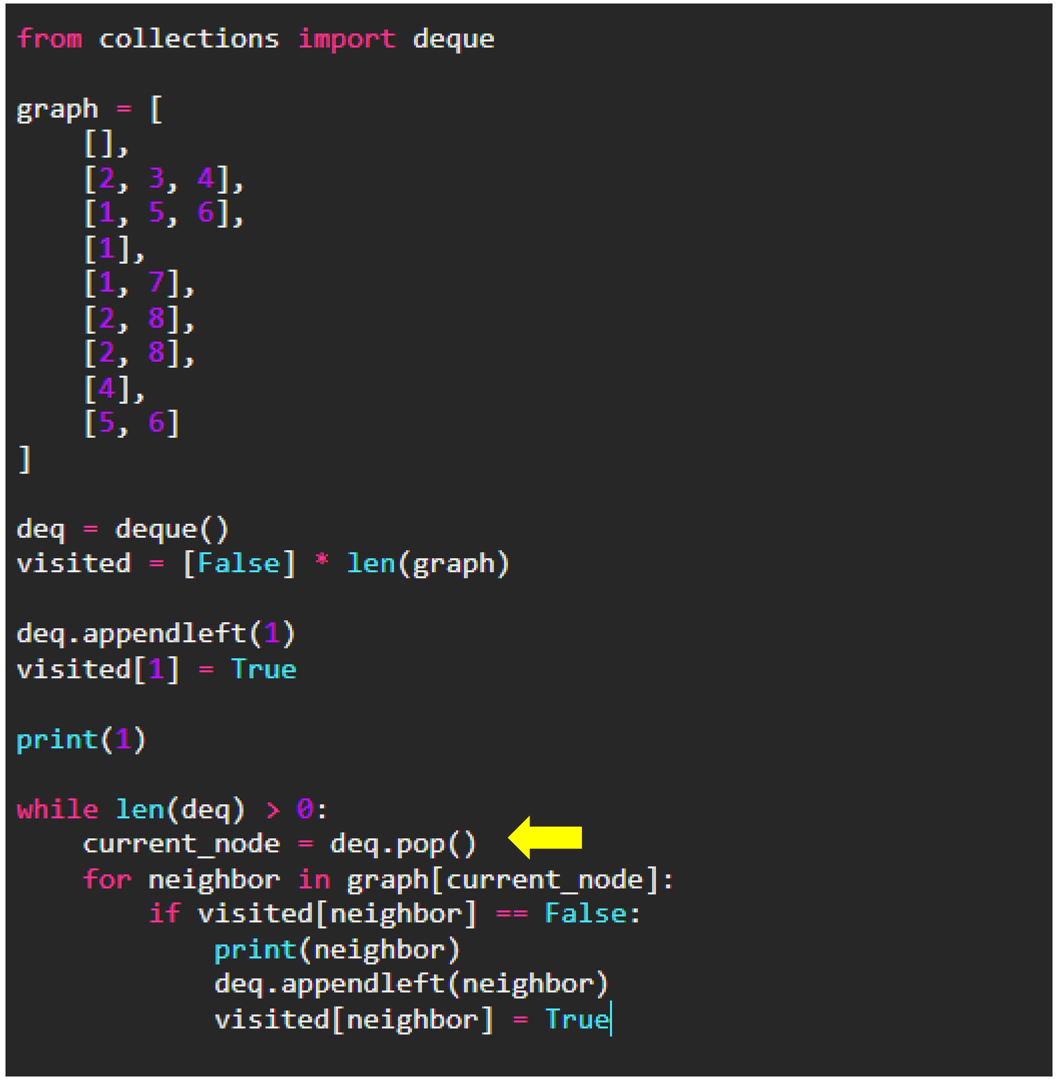




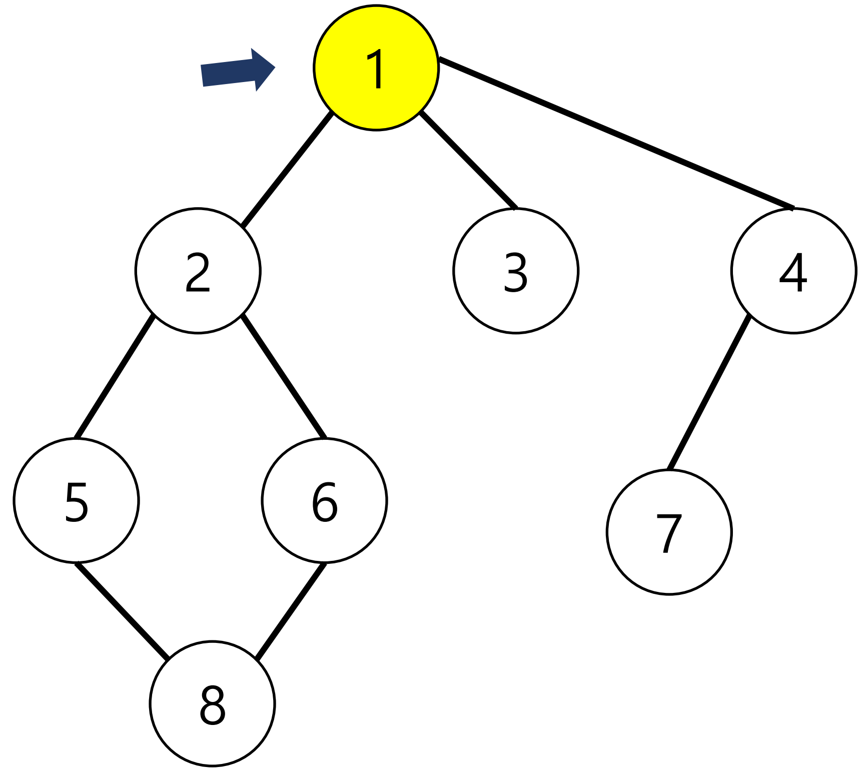
```python
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```

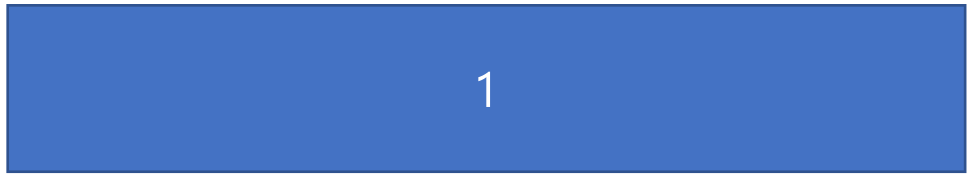


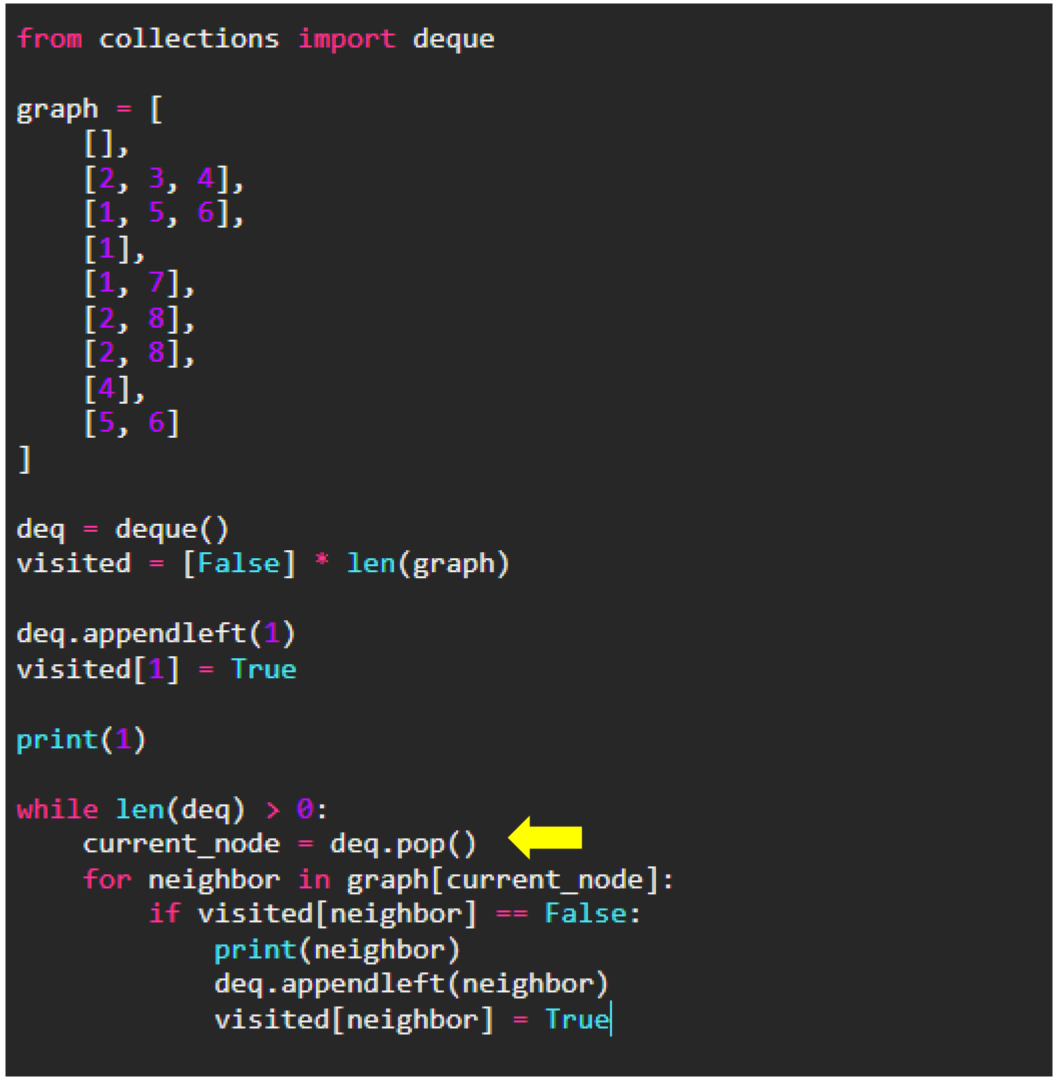




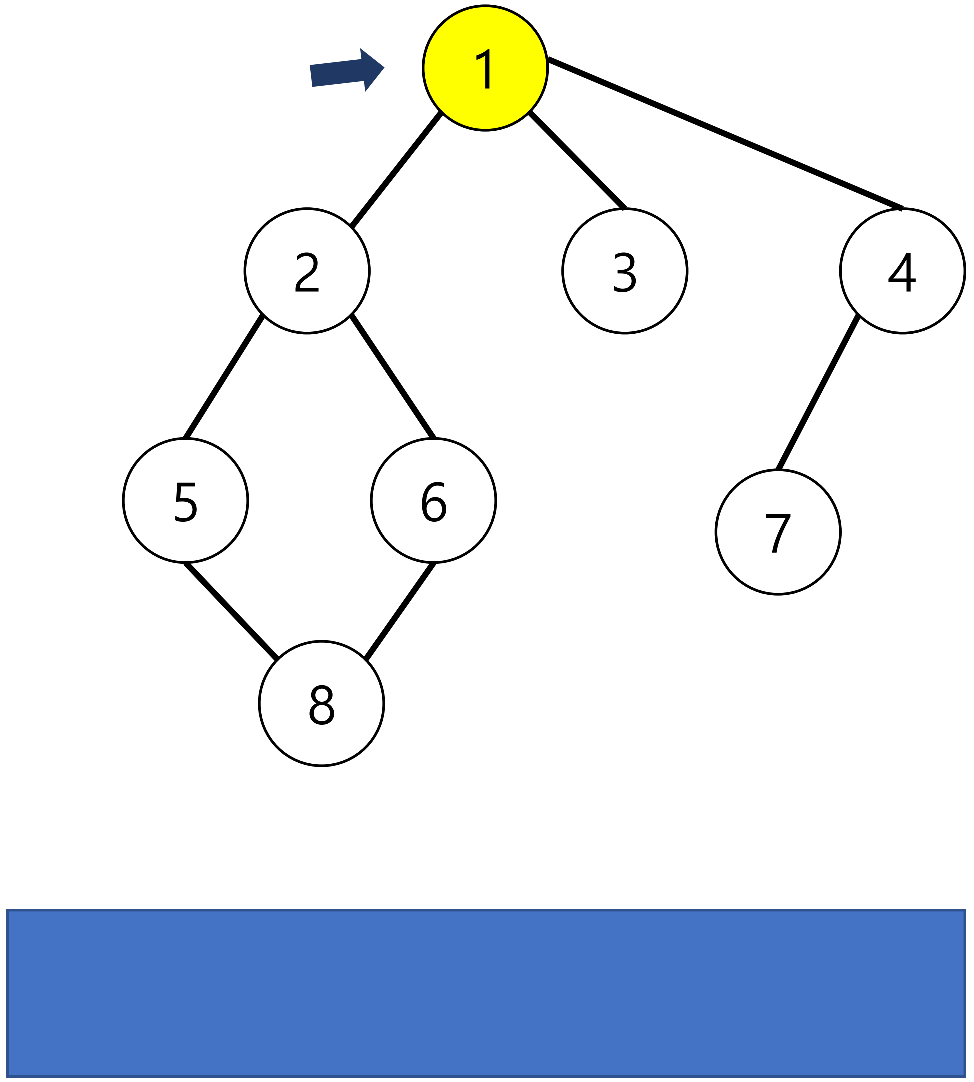
```
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```

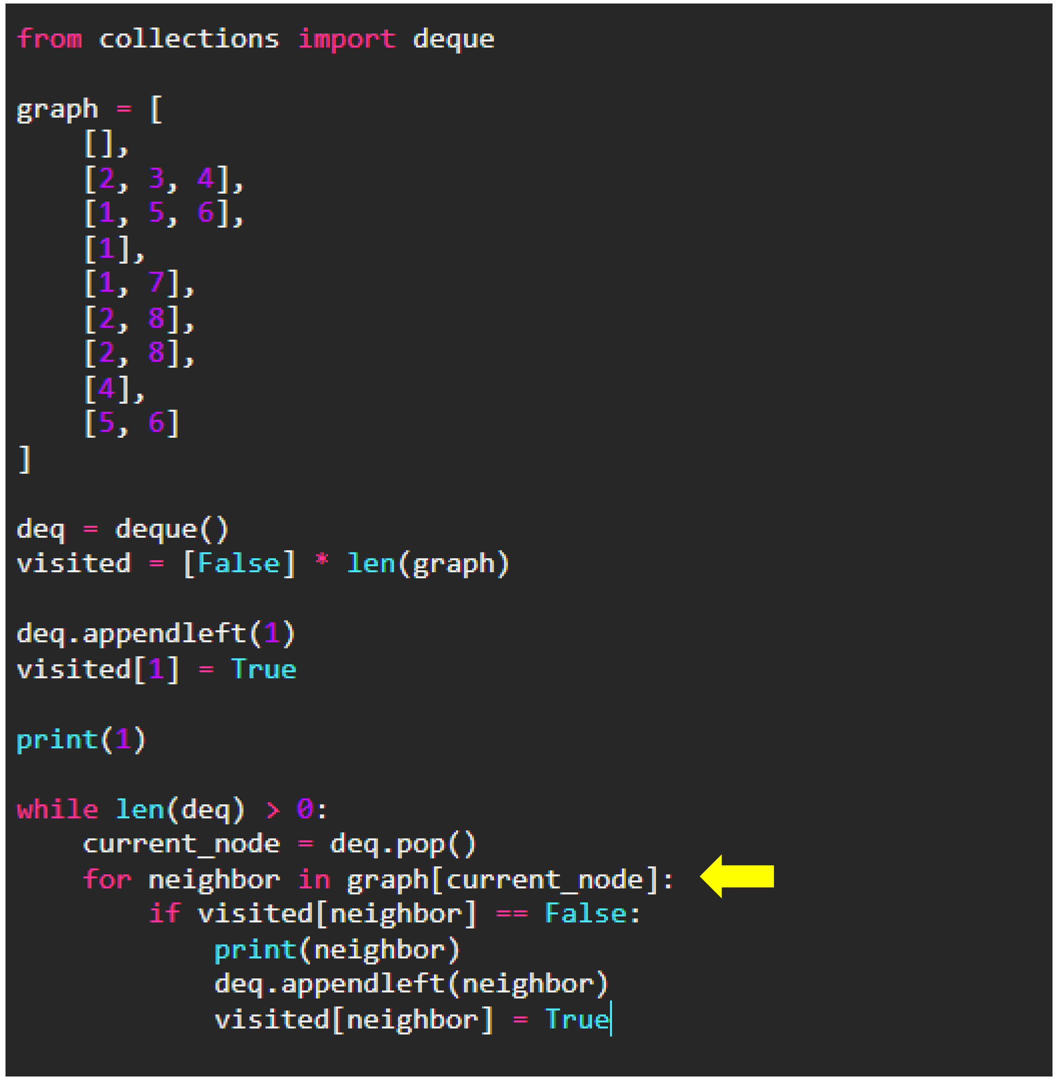


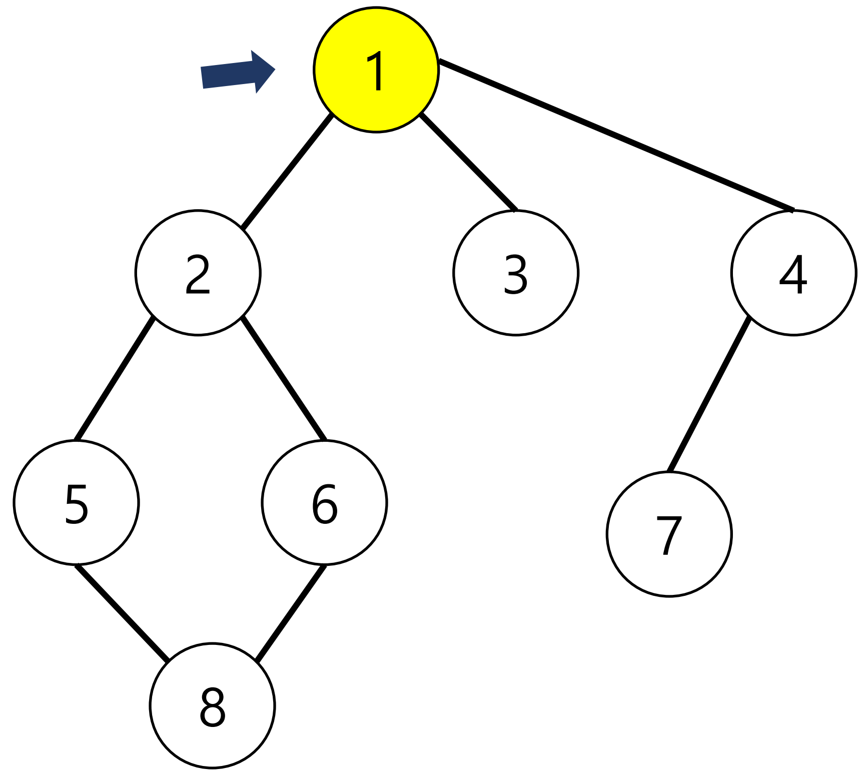




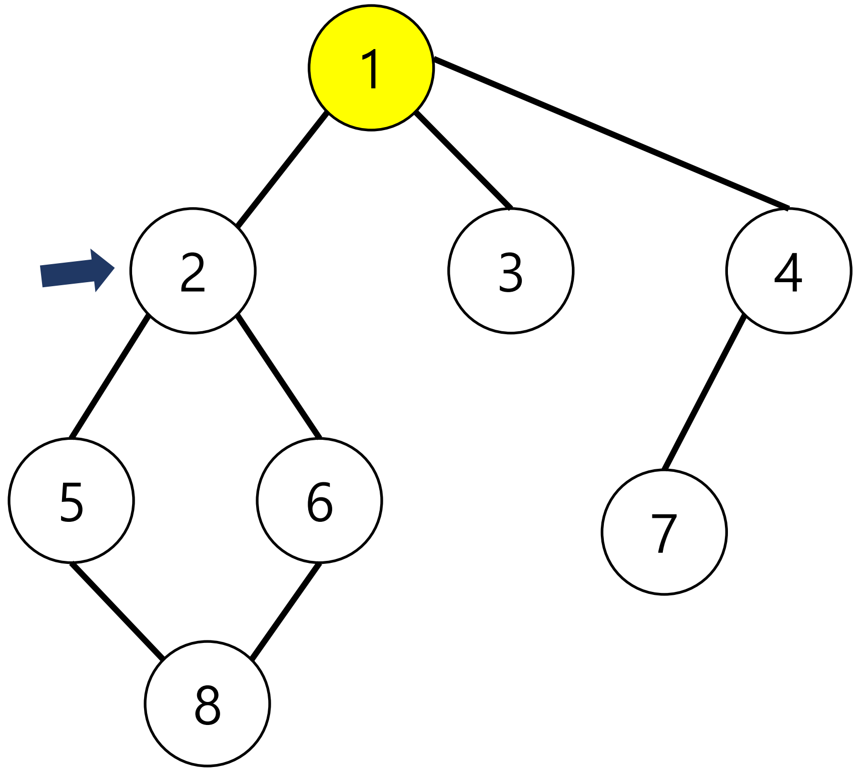
```
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```

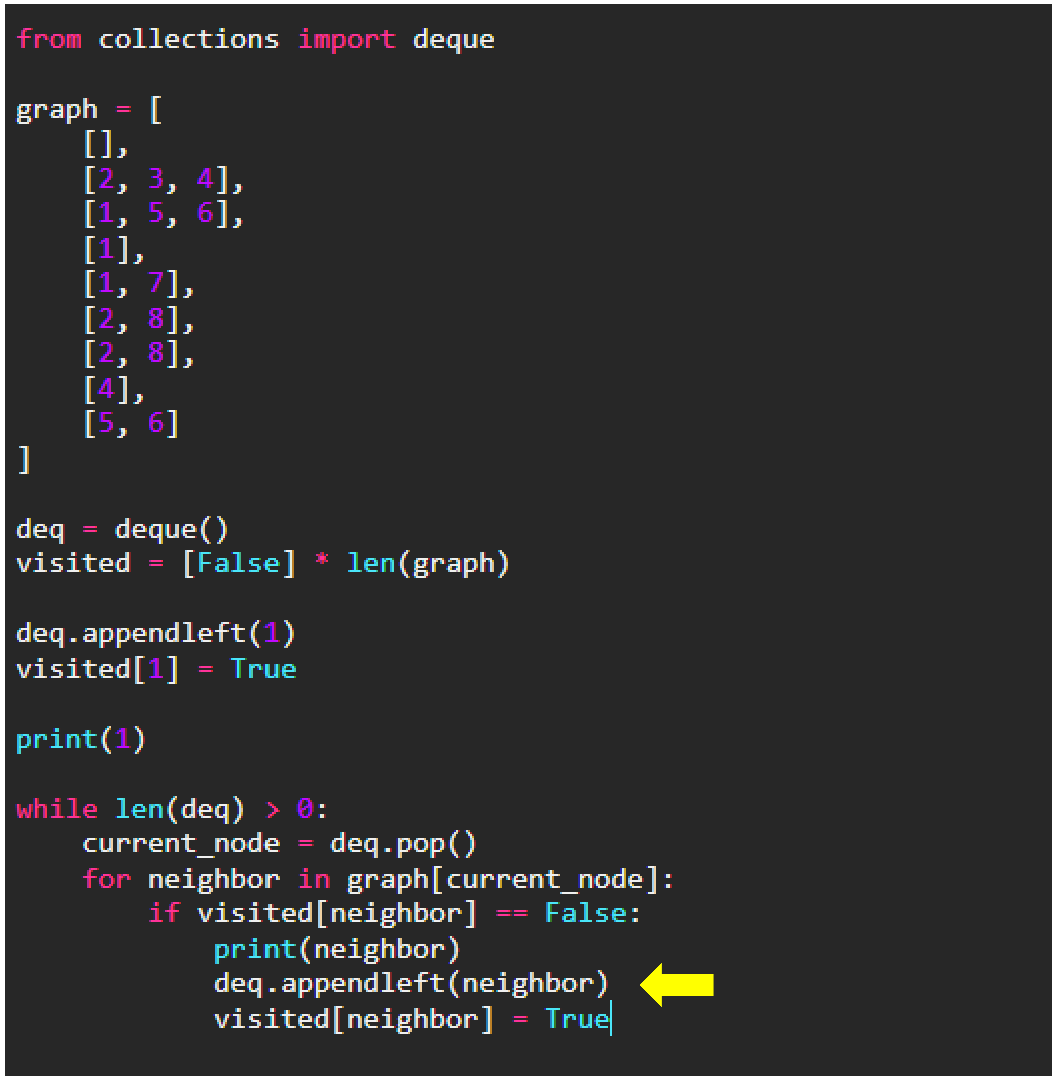



```
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```

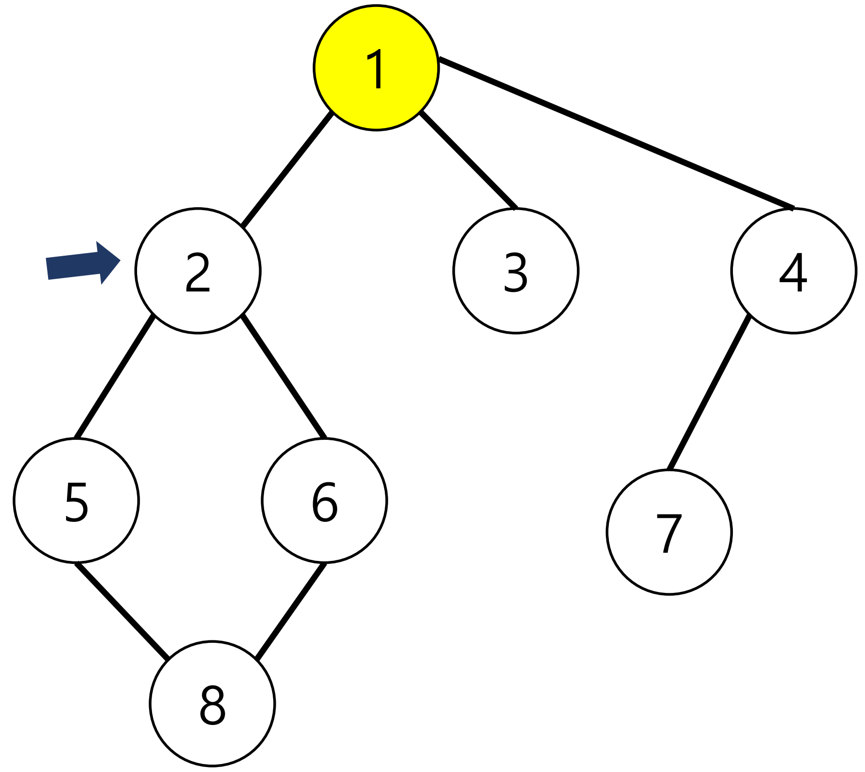


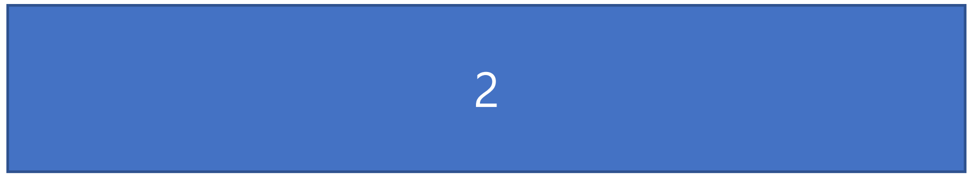
```
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```

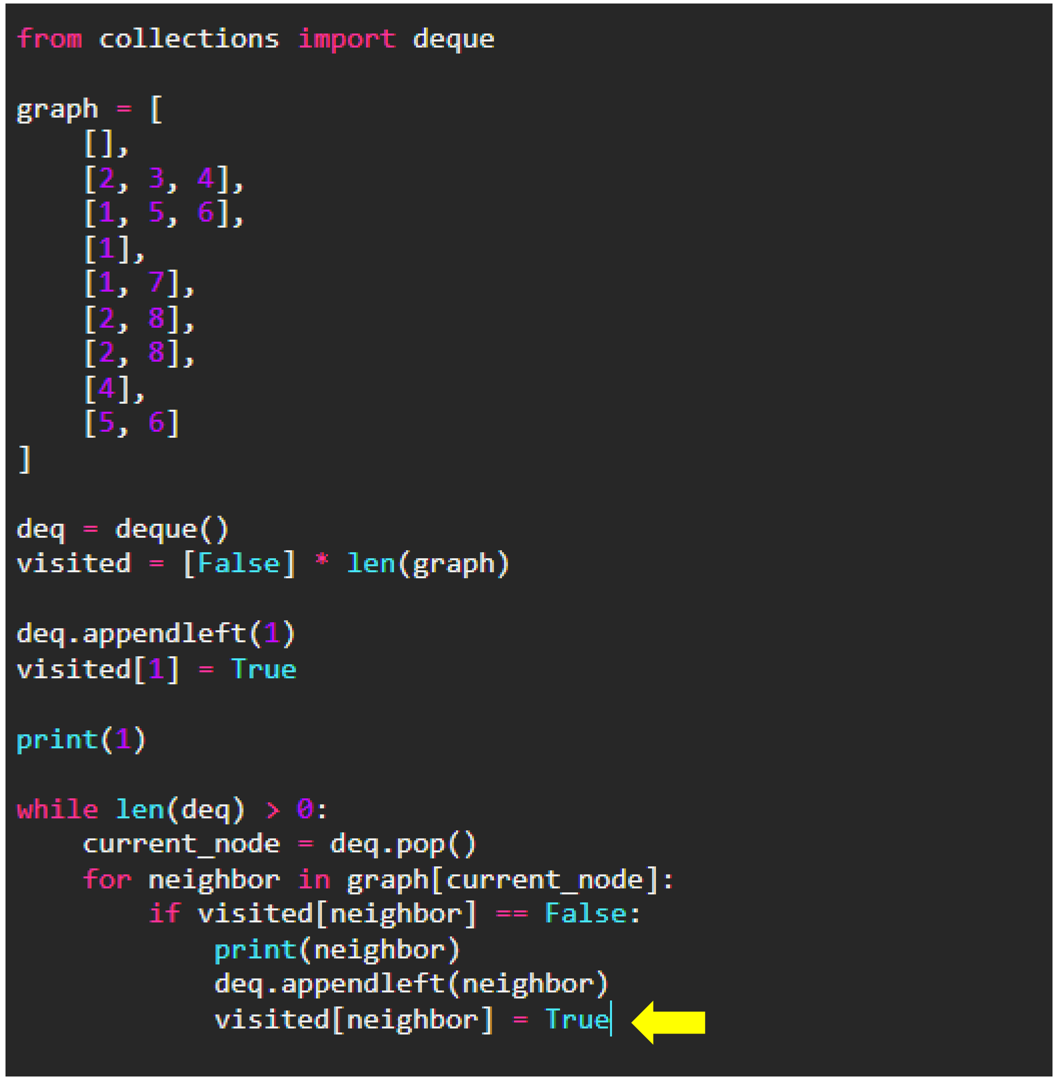




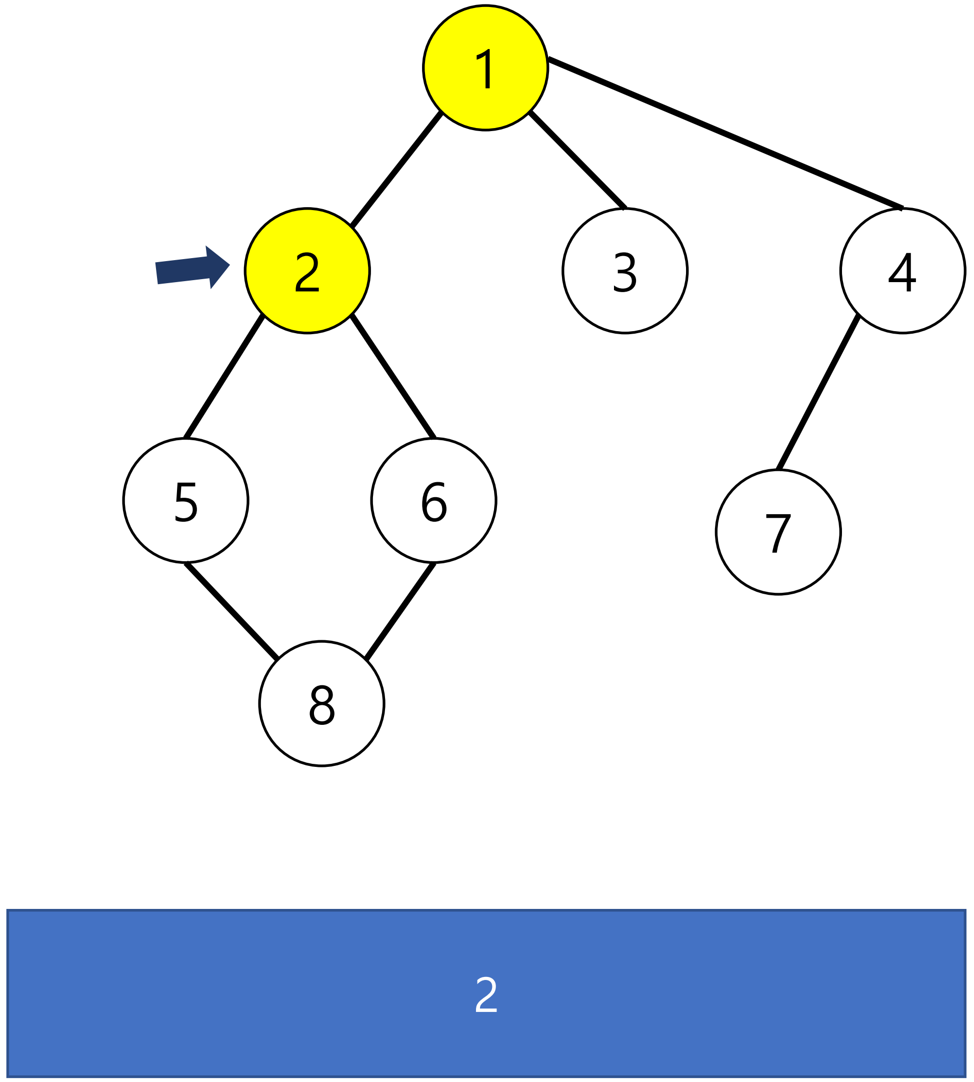
```
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```

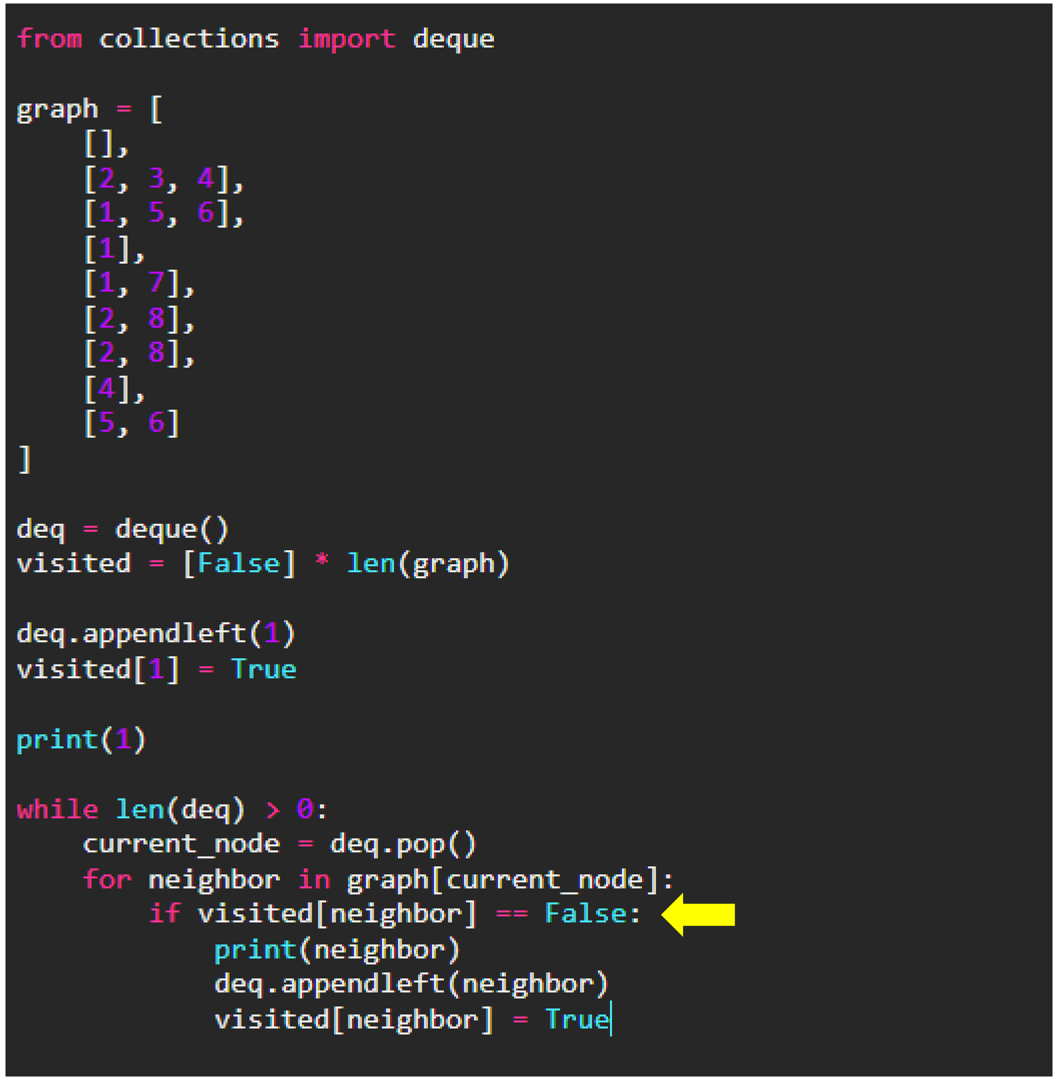




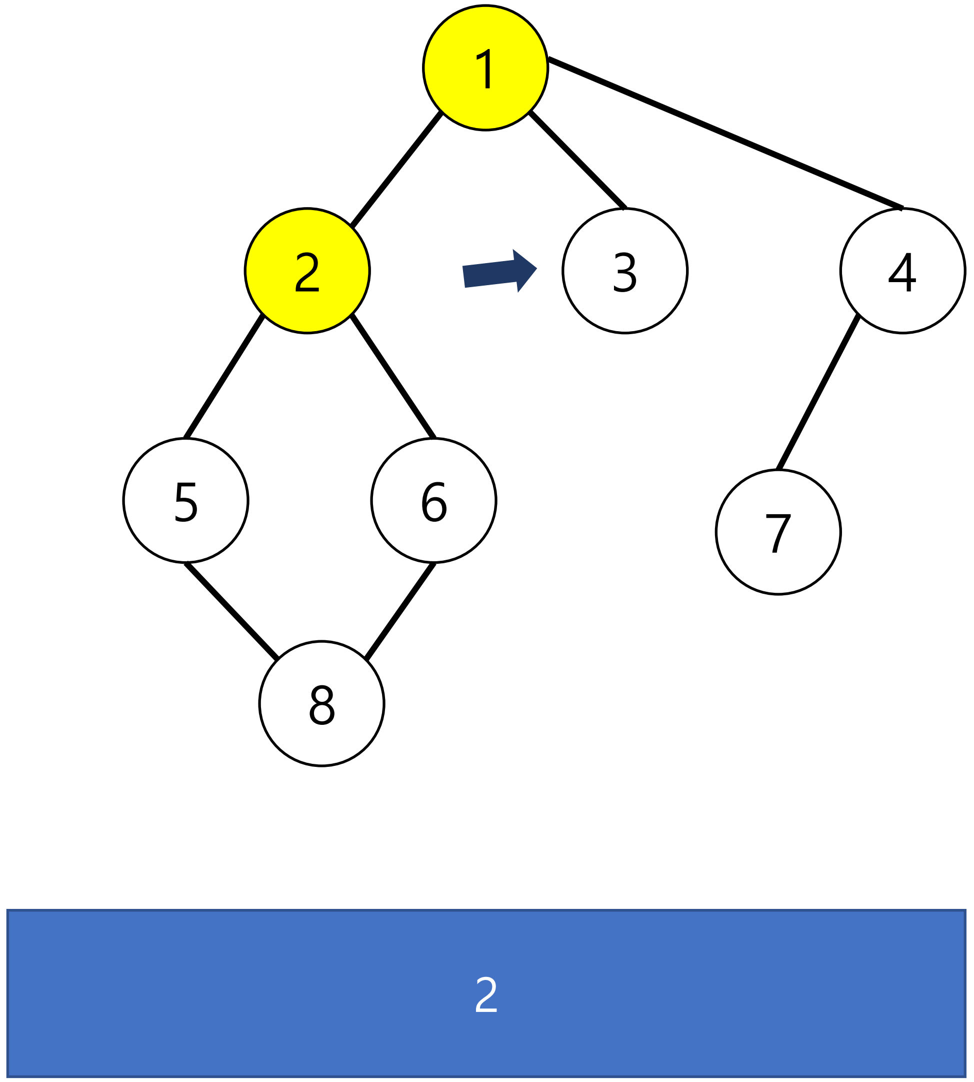


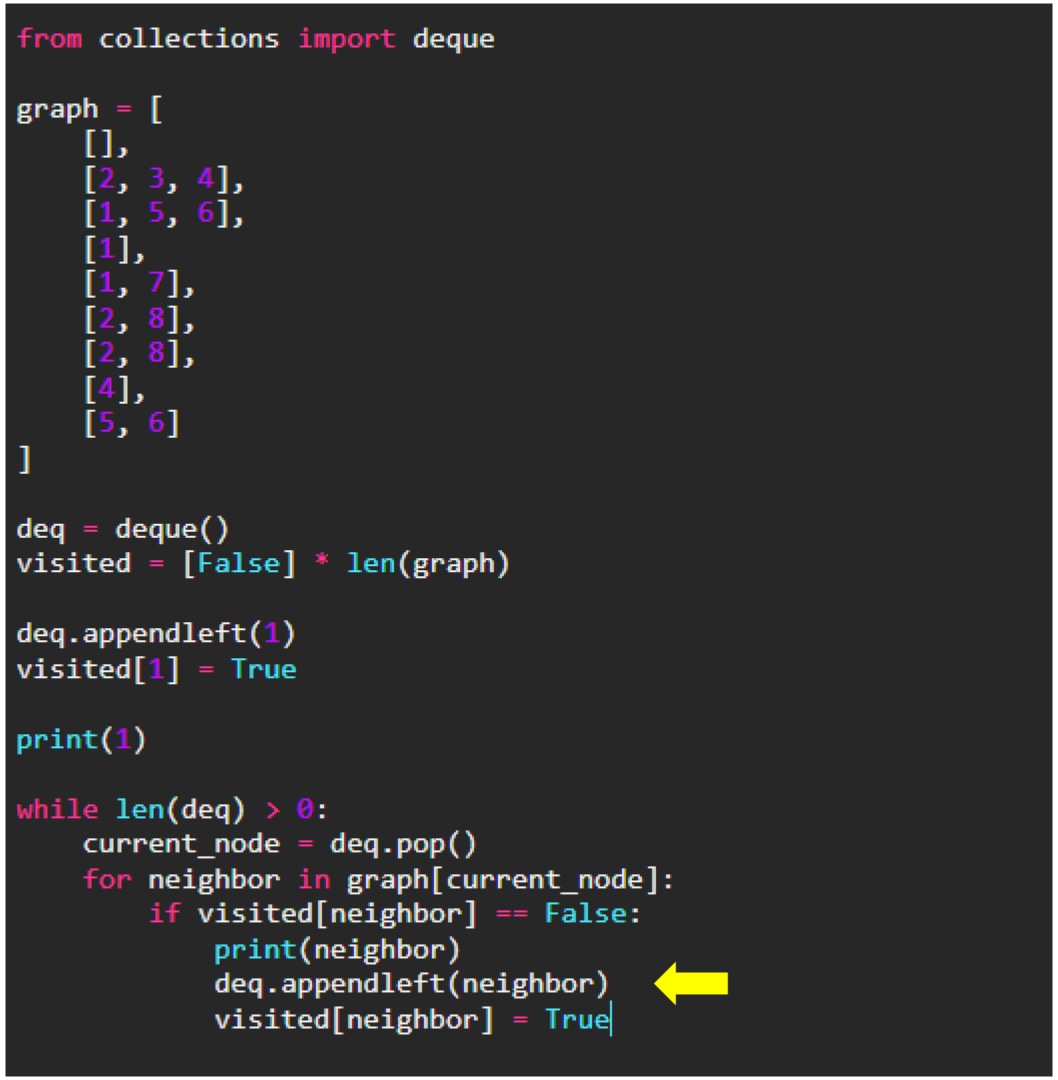
```
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```



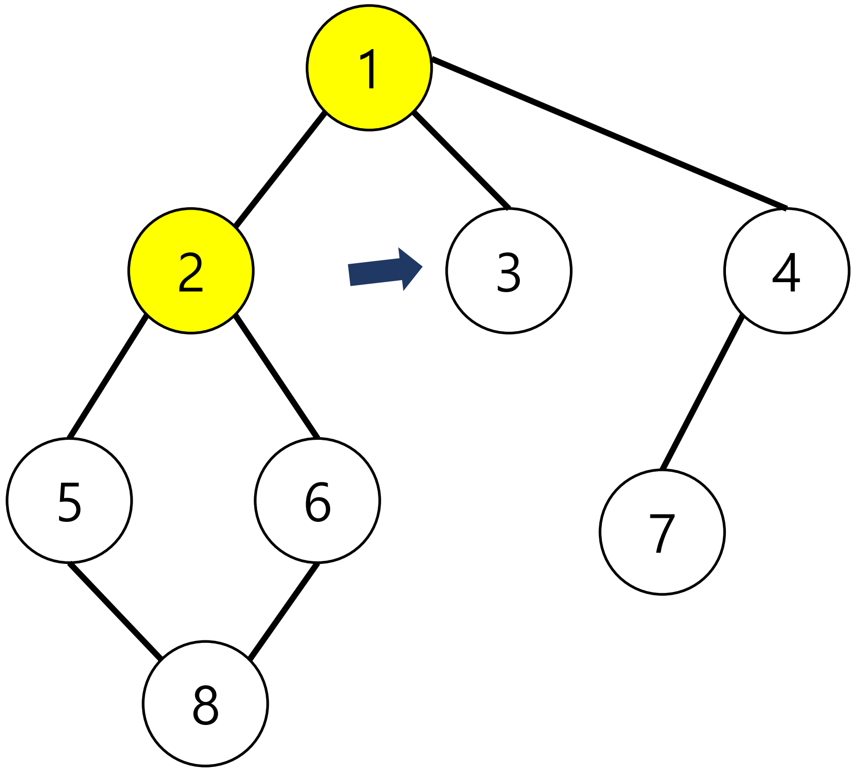


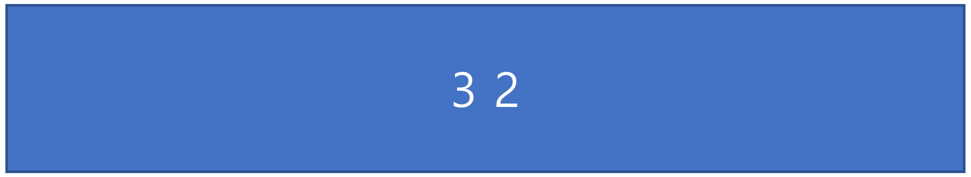
```
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```

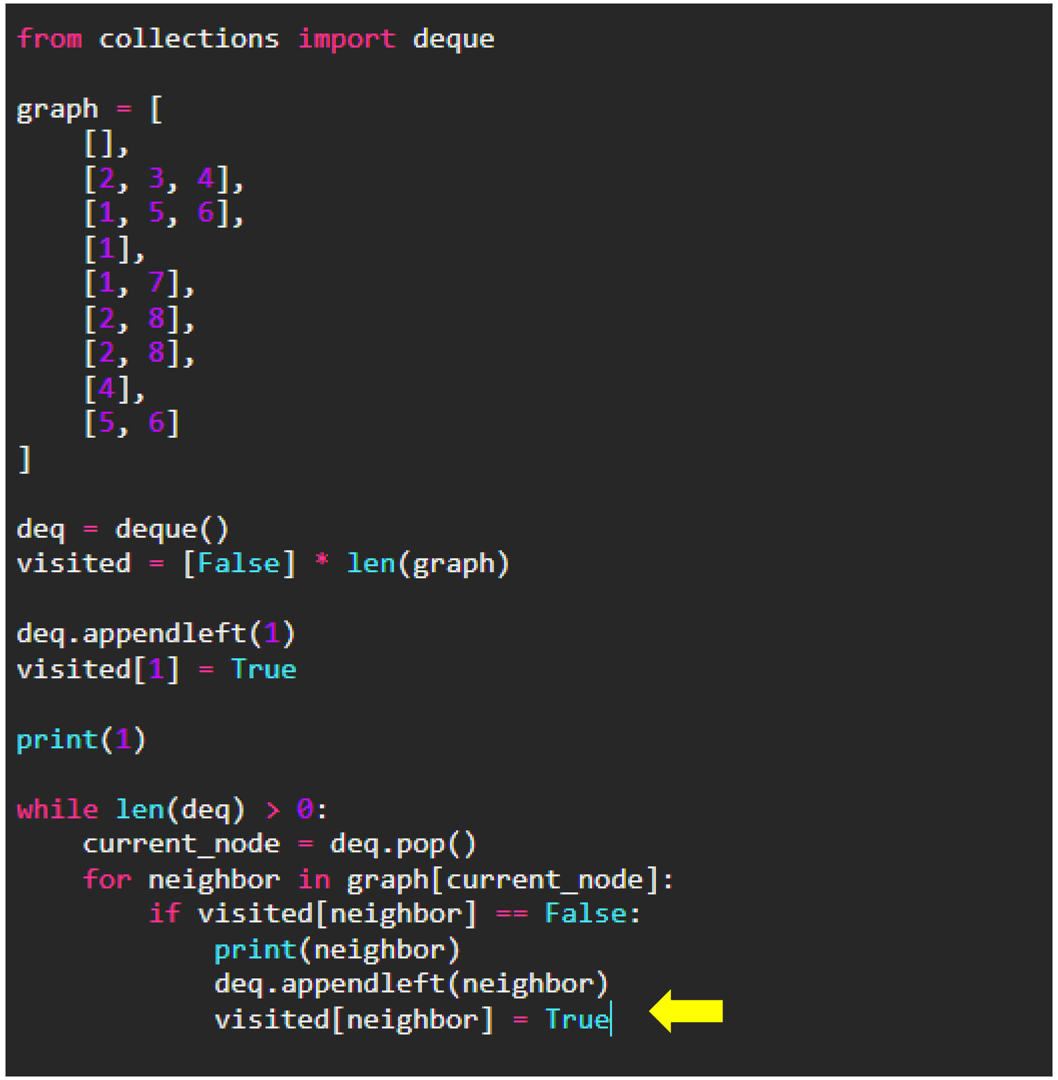




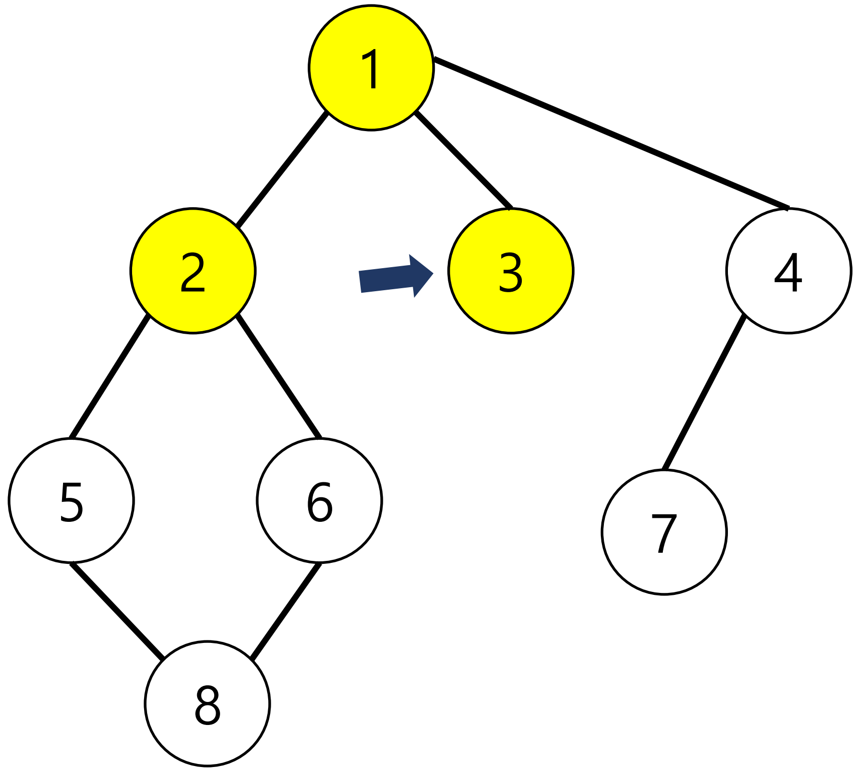
```
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```



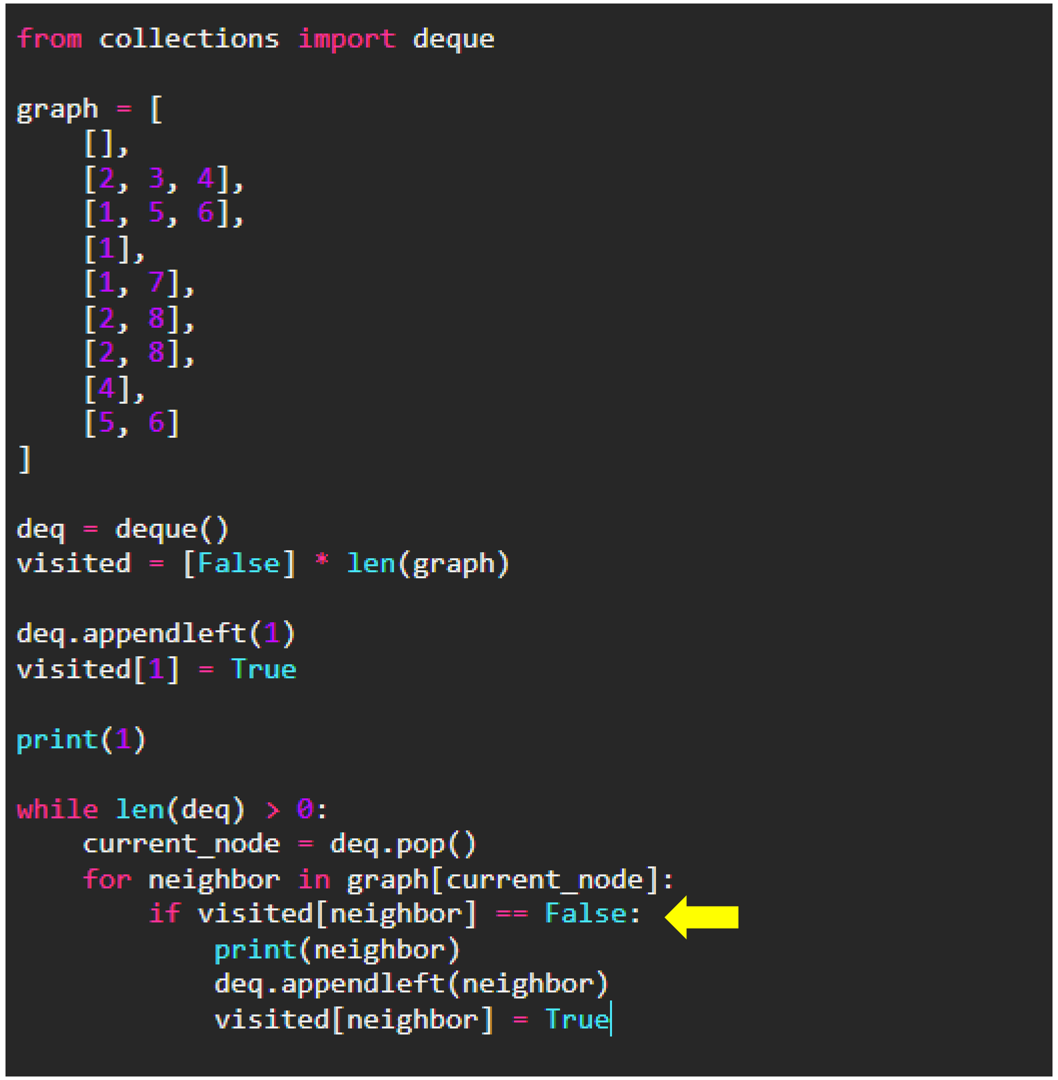


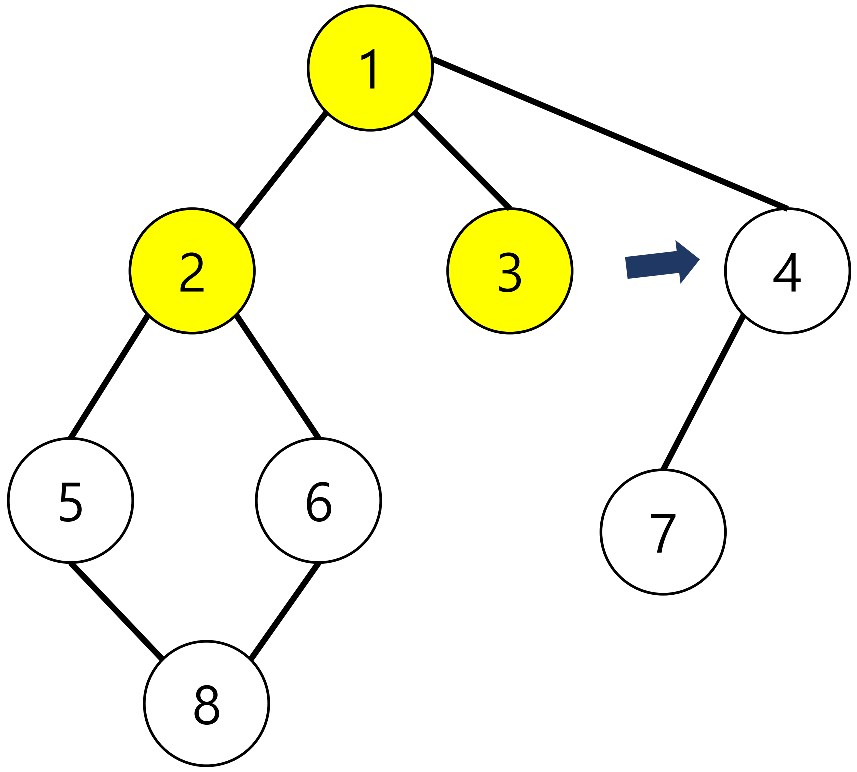


```
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```

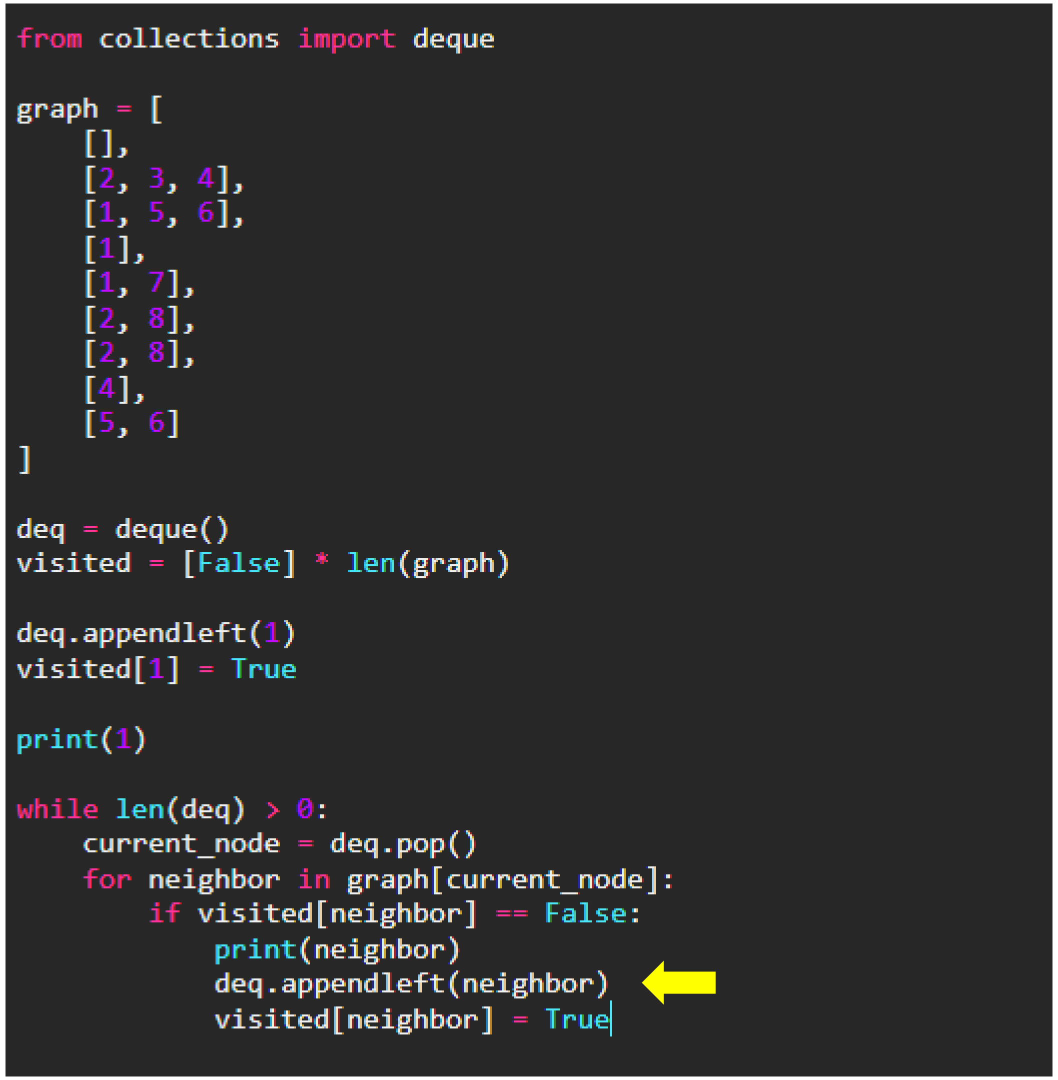


3 2

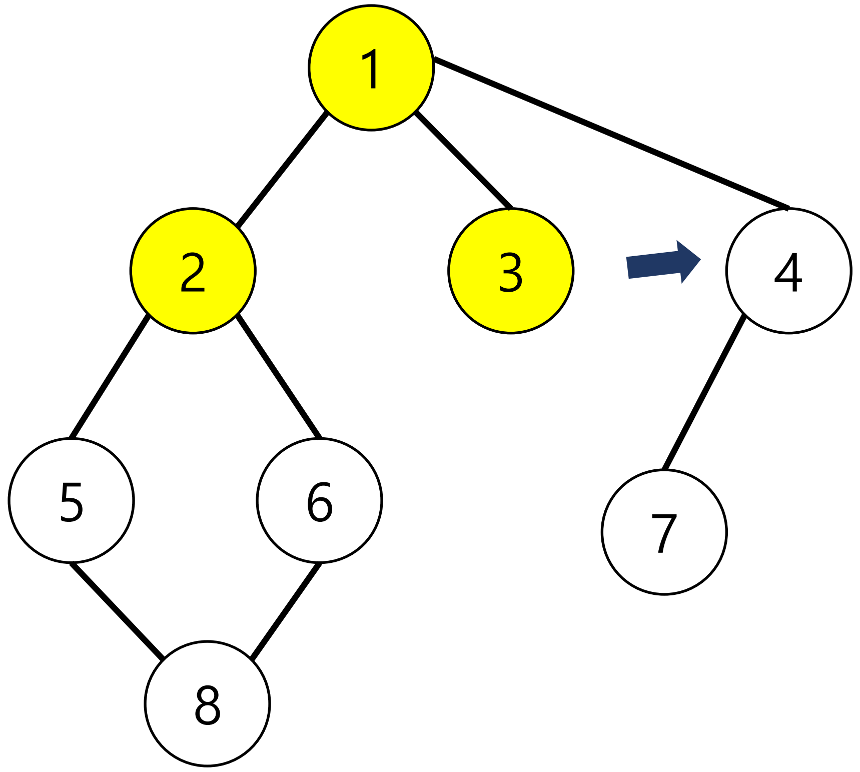

```
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```



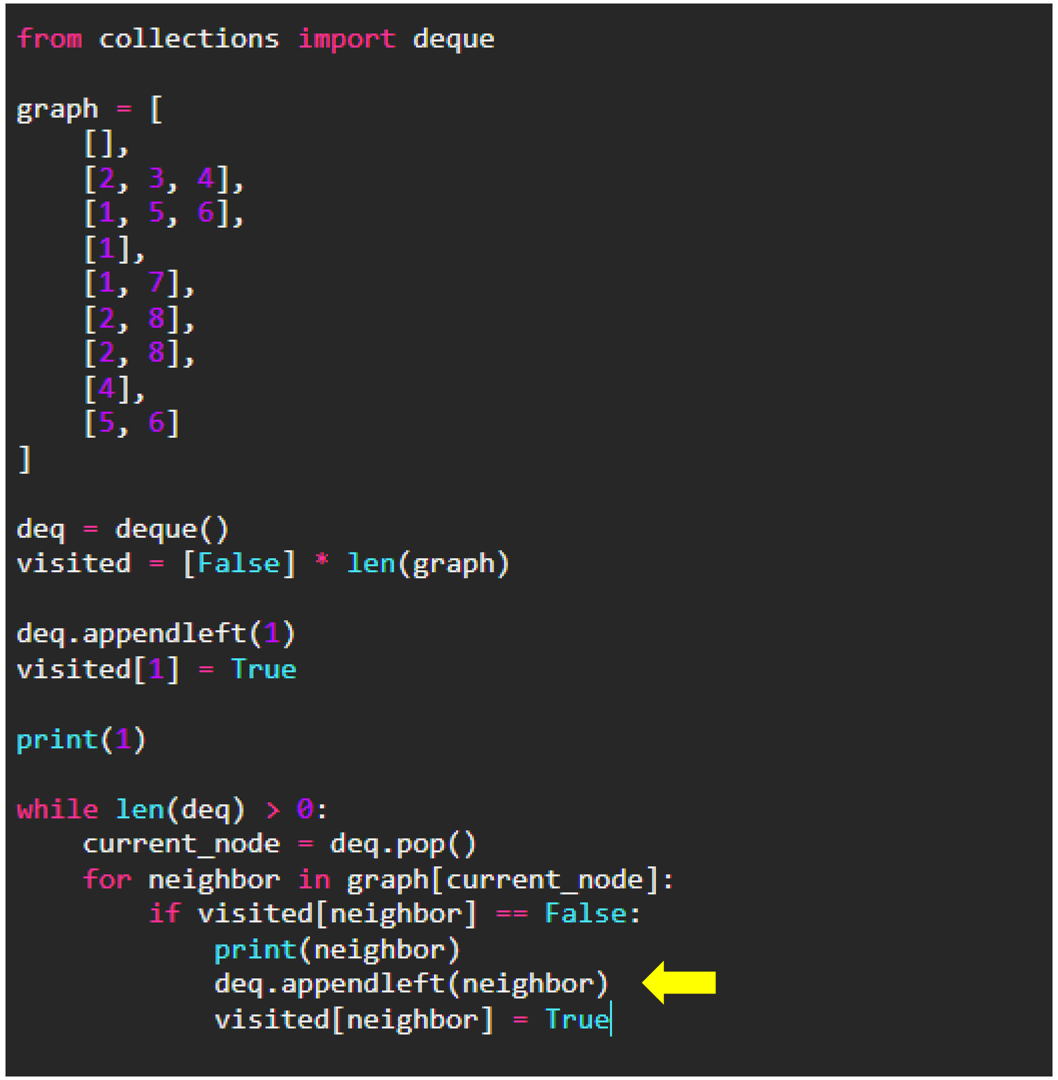
3 2



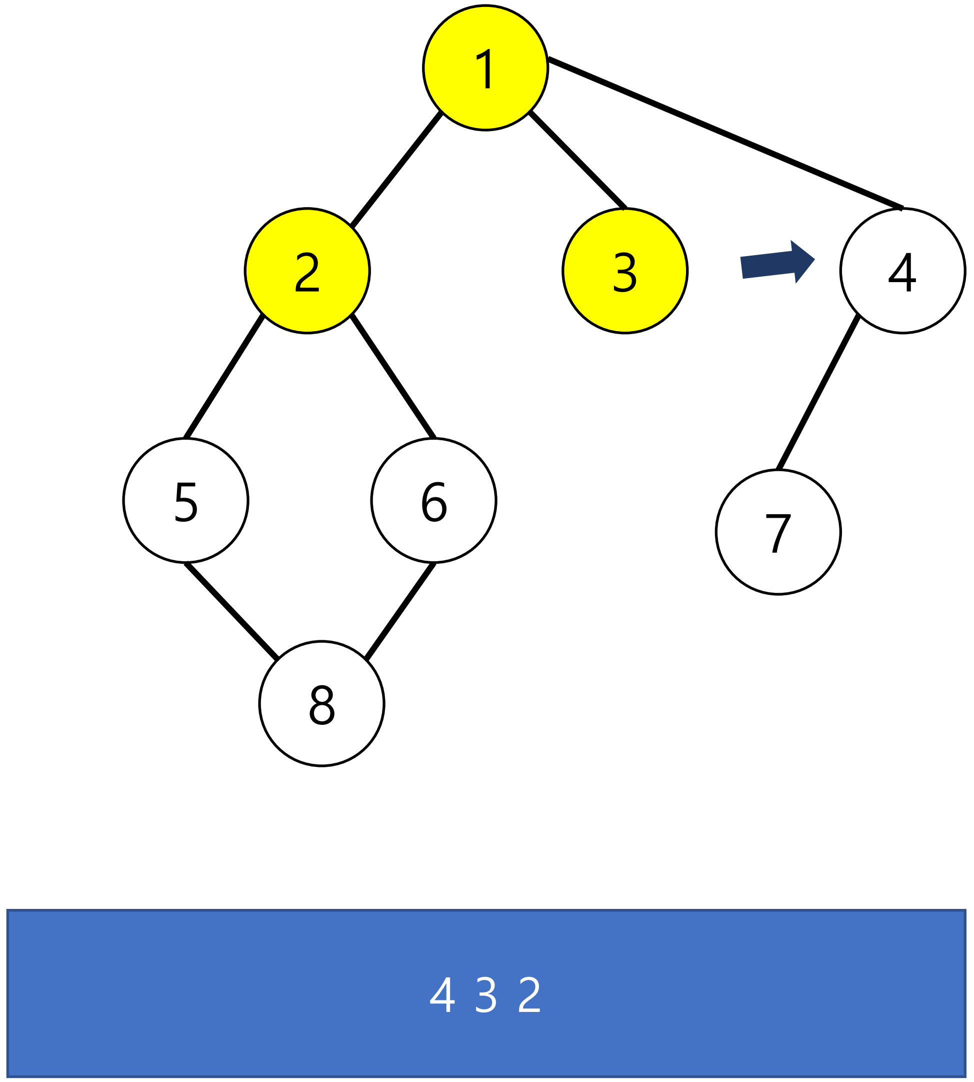
```
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```



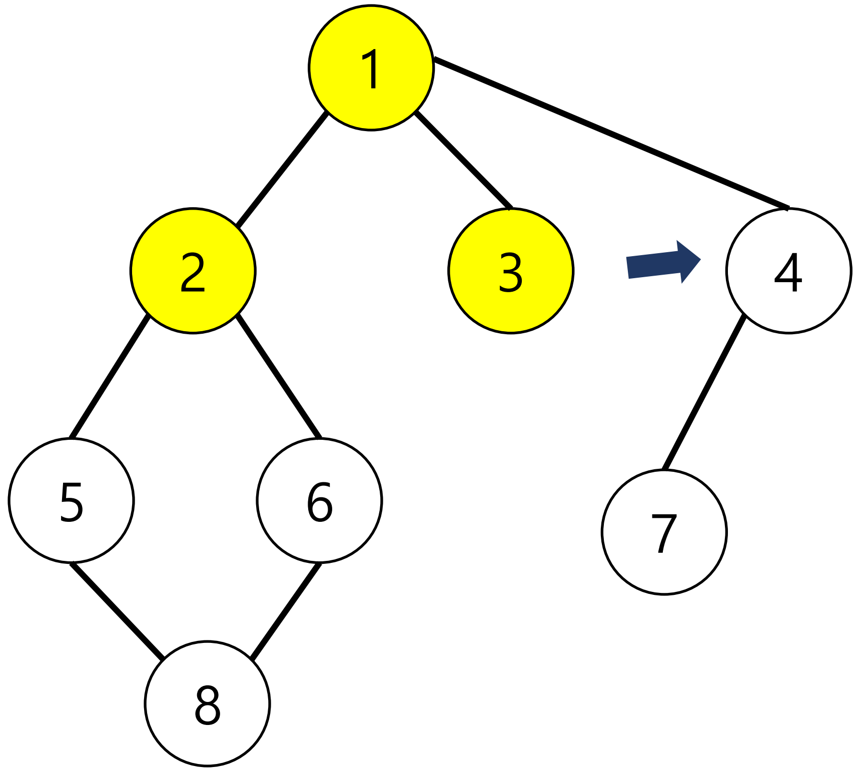
3 2



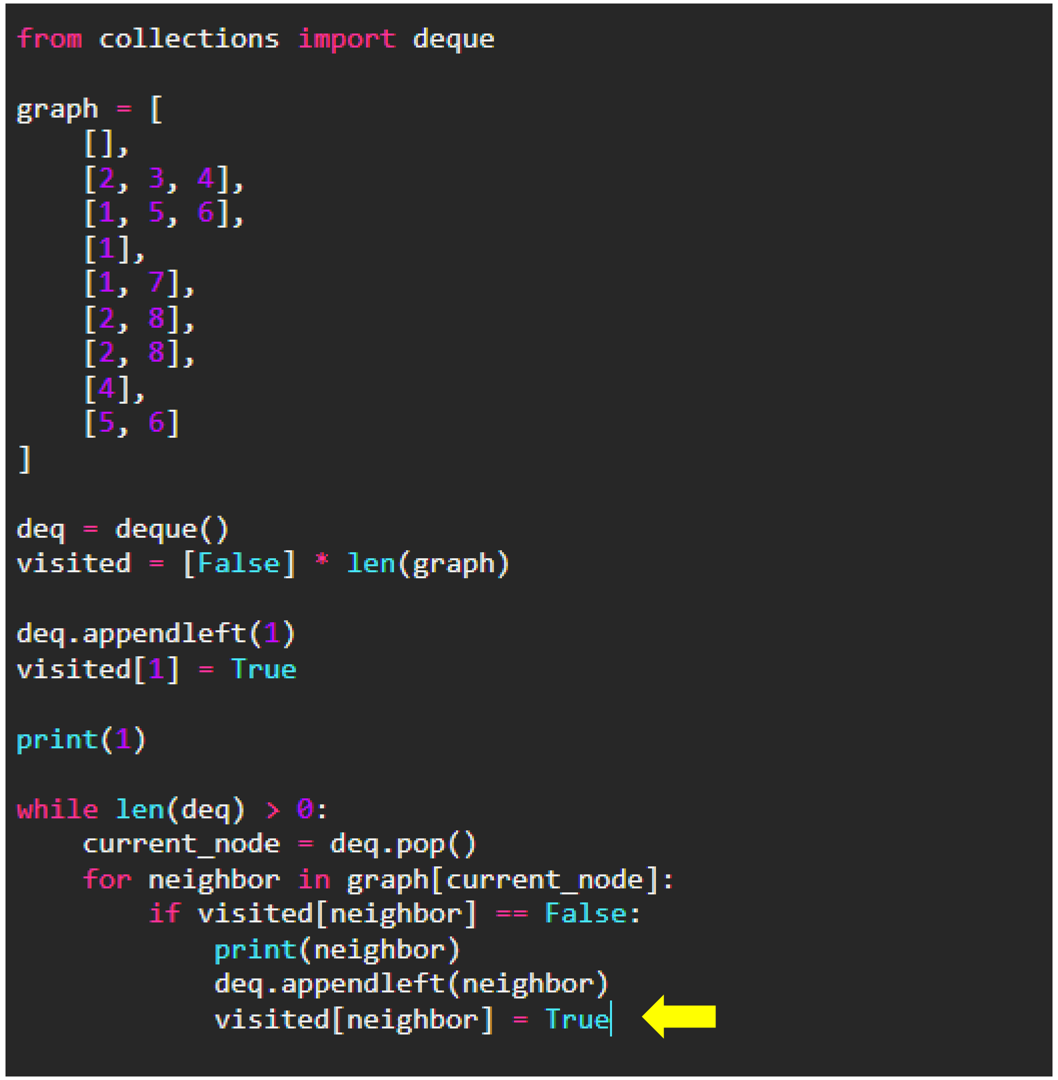
```
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```



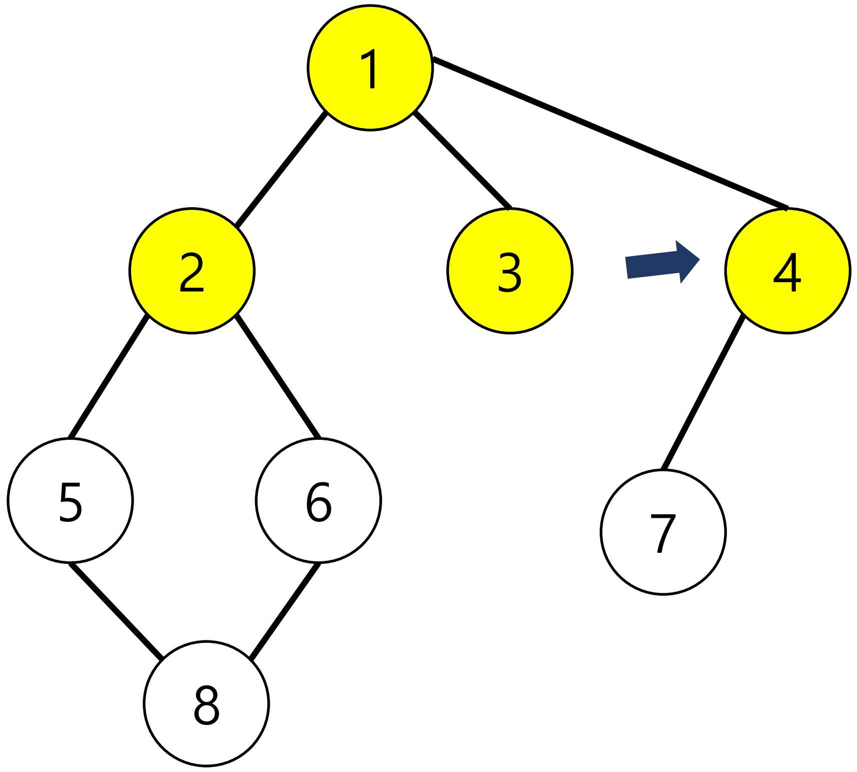
```python
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```

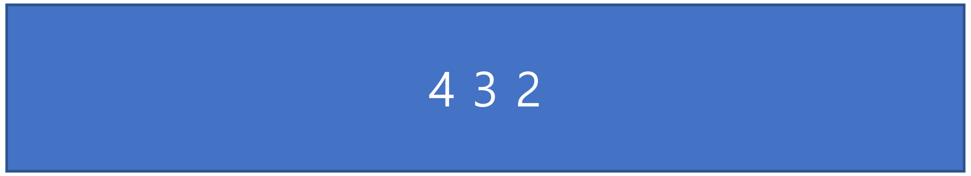


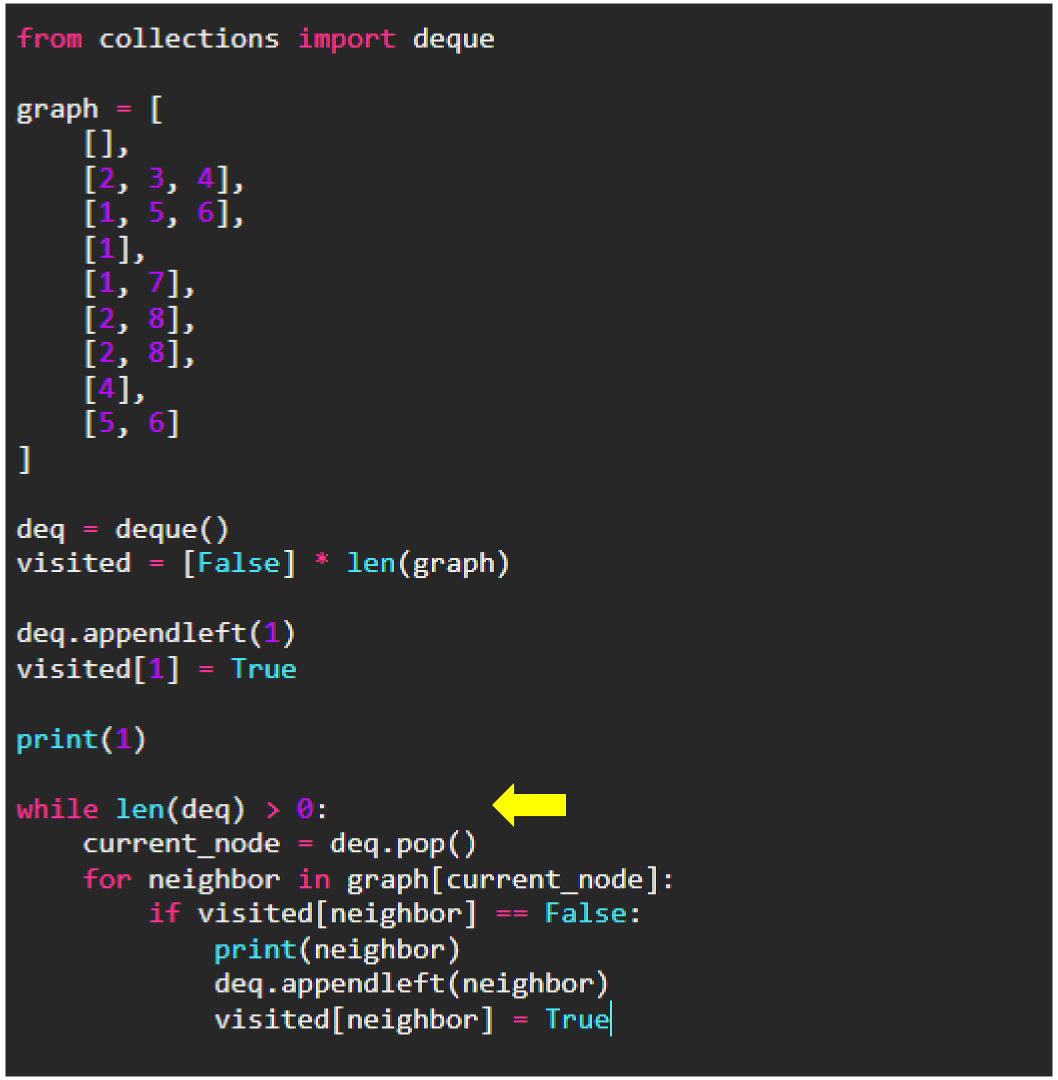
4 3 2



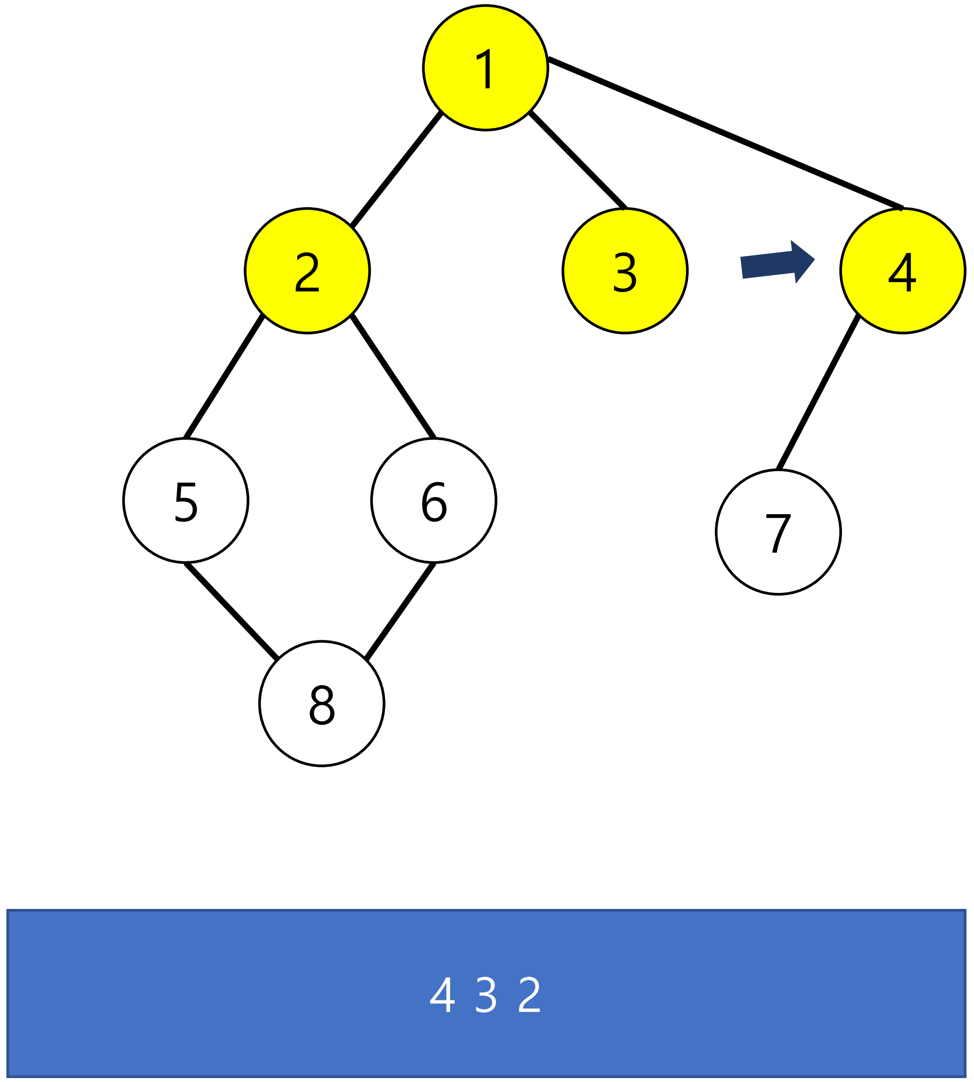
```
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```

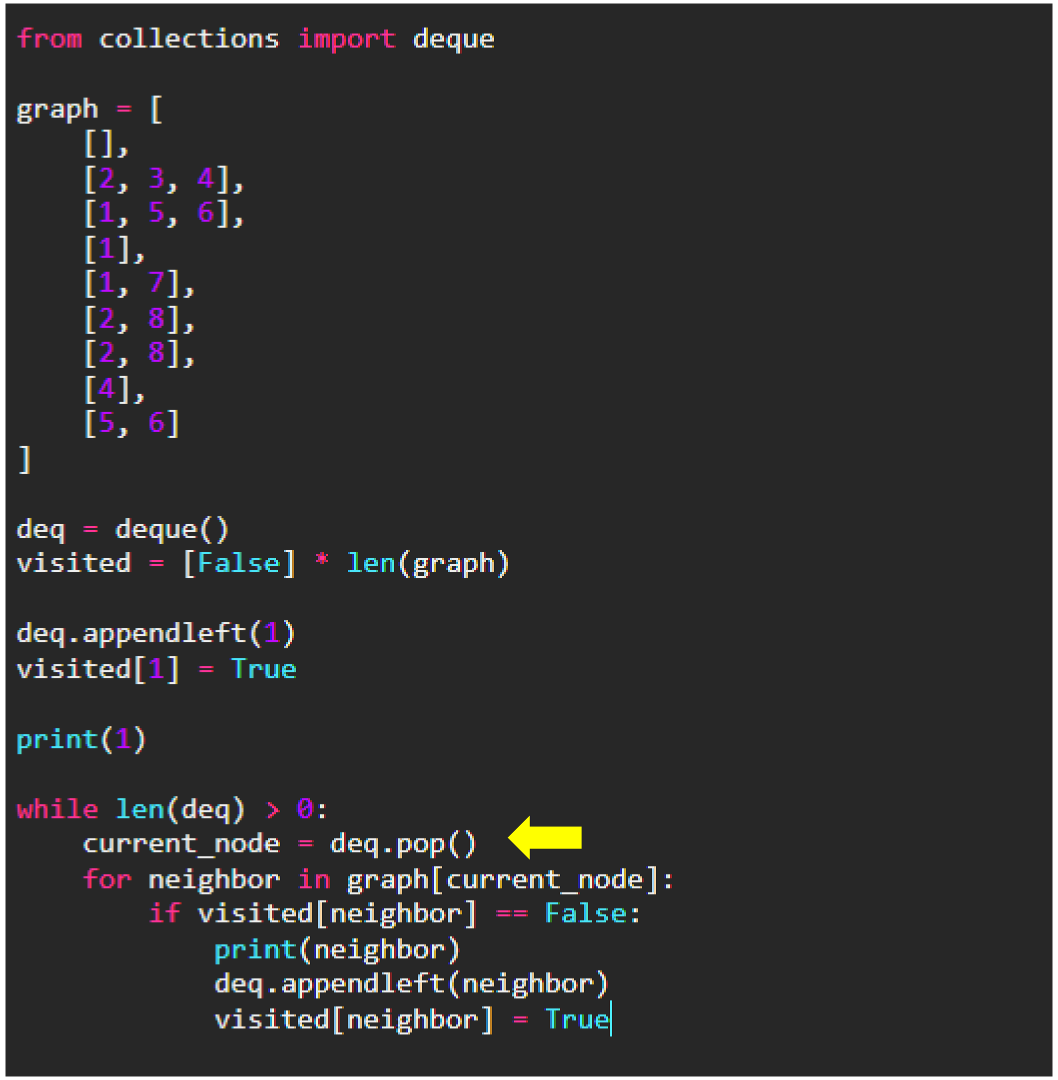




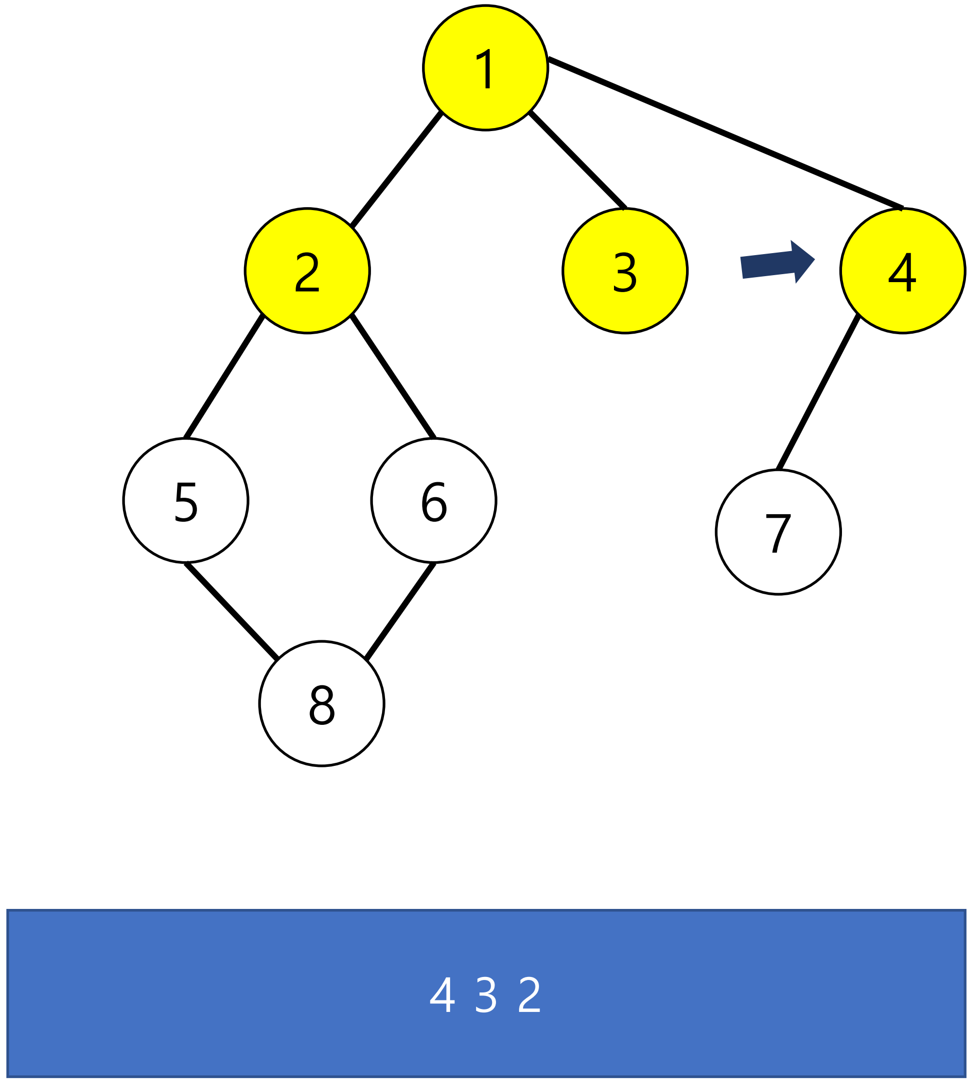


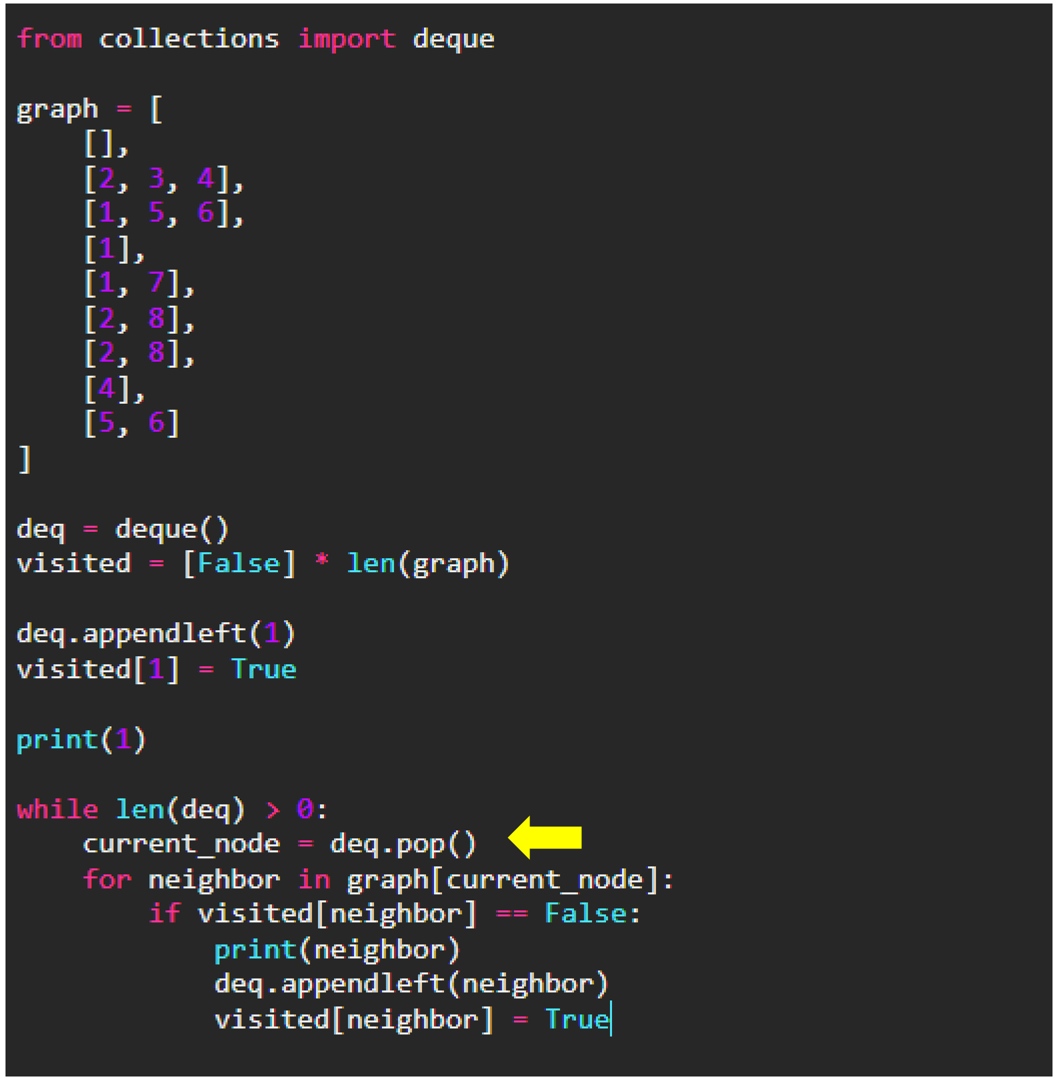
```
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```



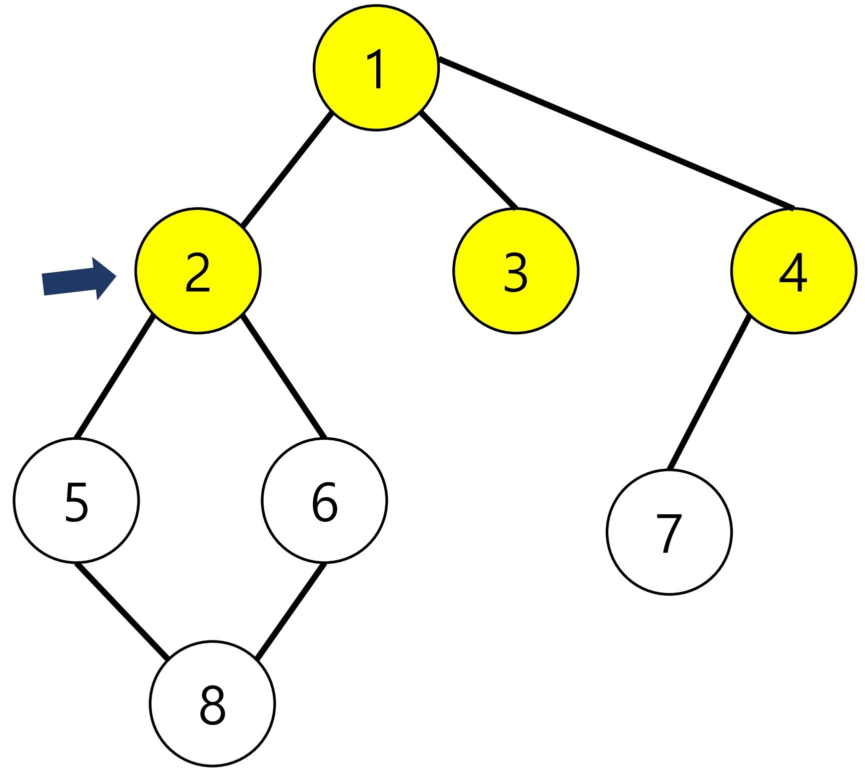


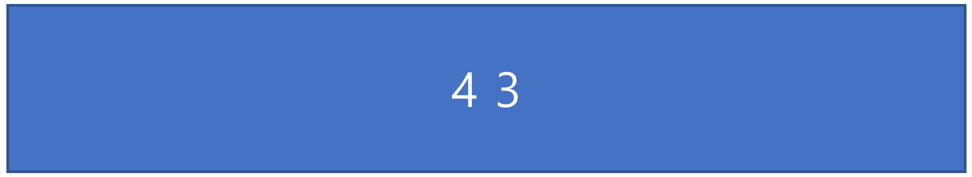
```
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```

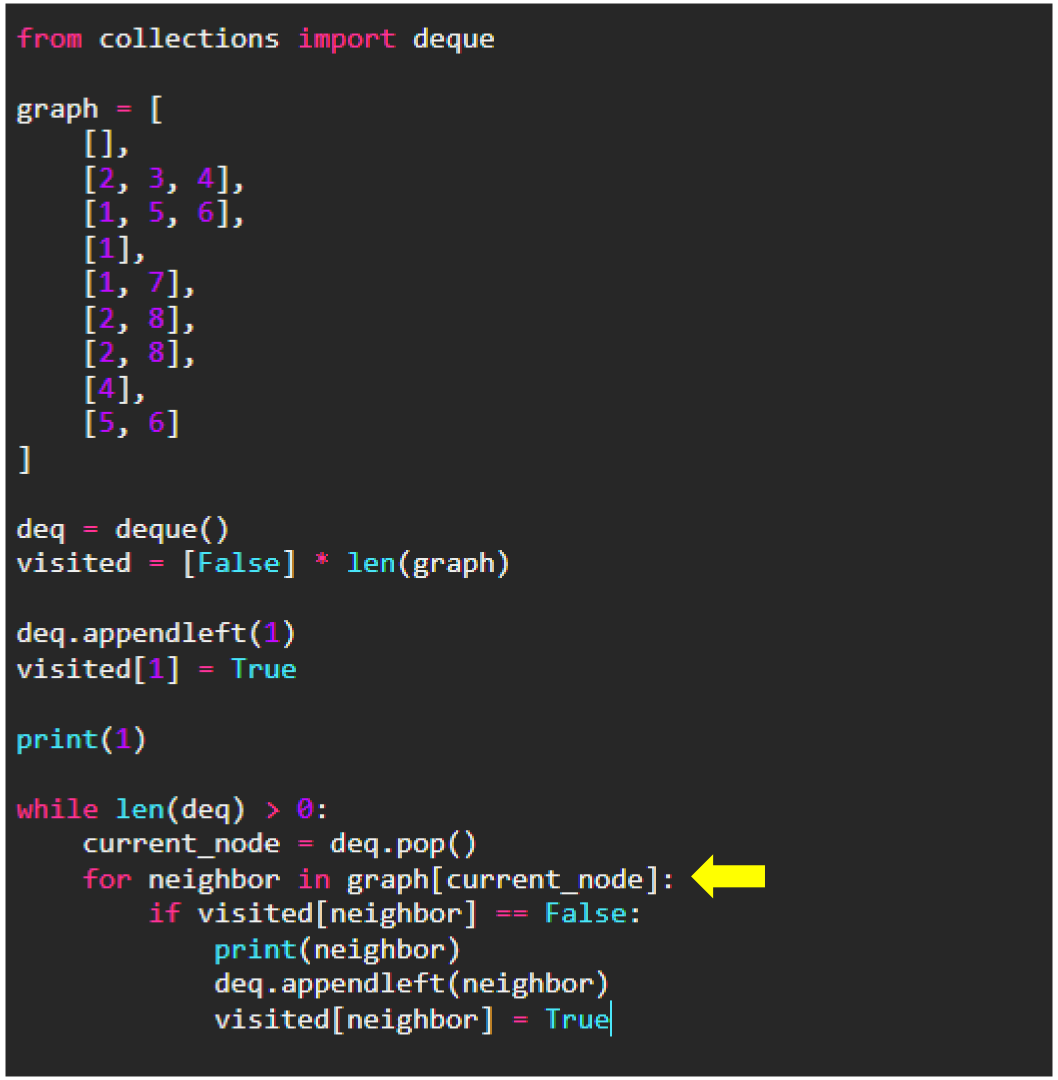




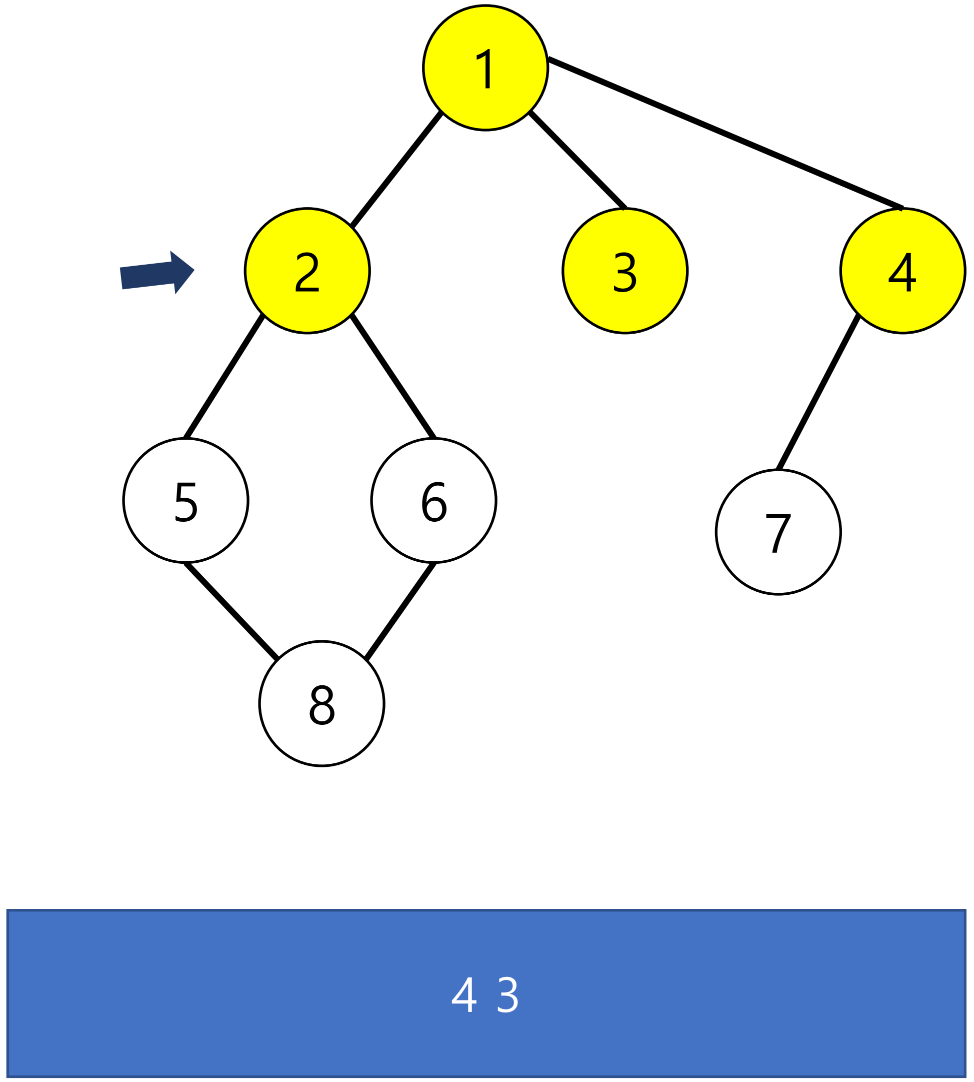
```
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```



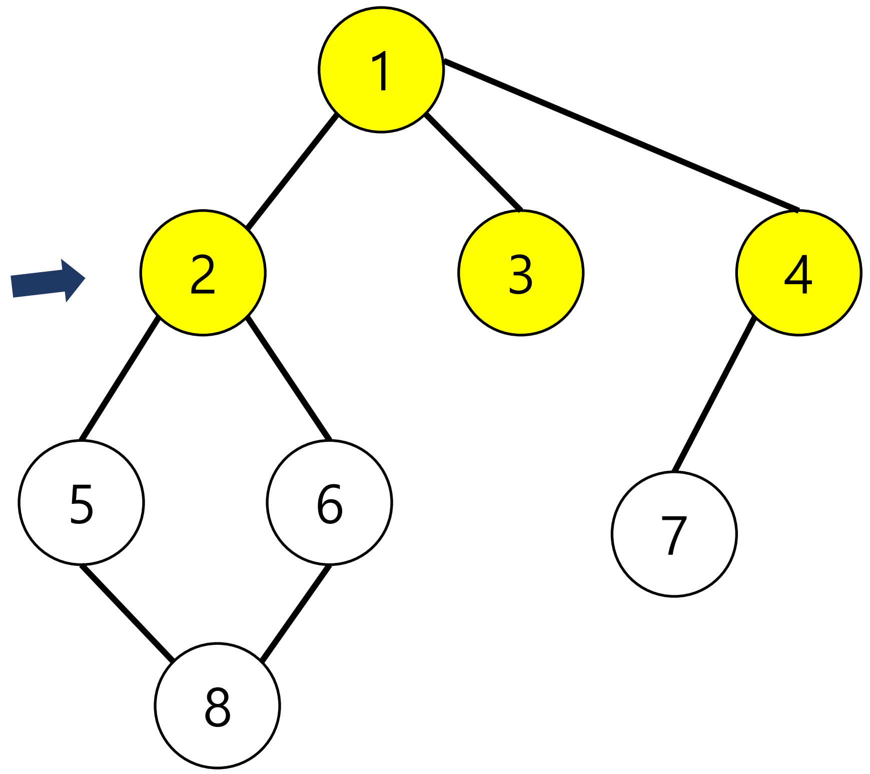




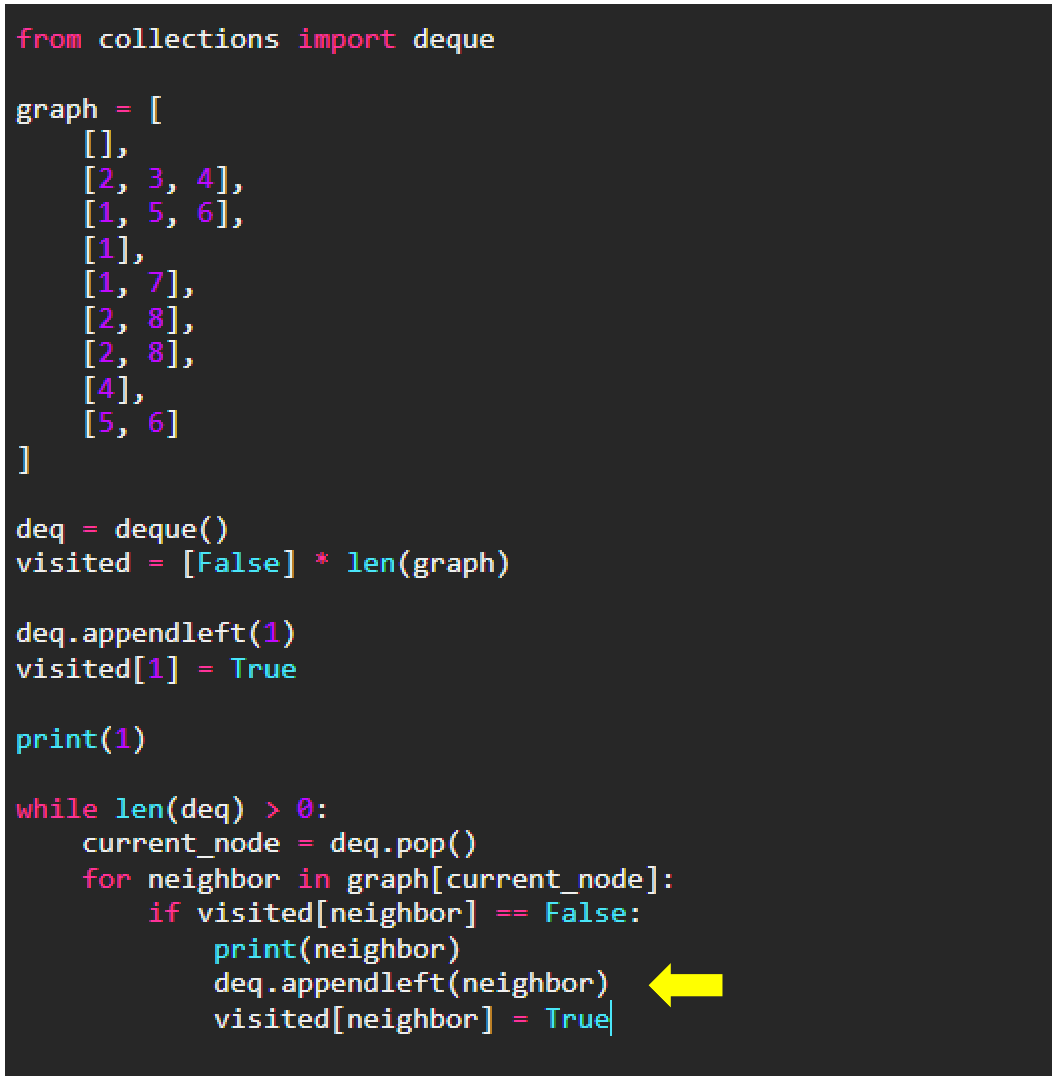
```python
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```



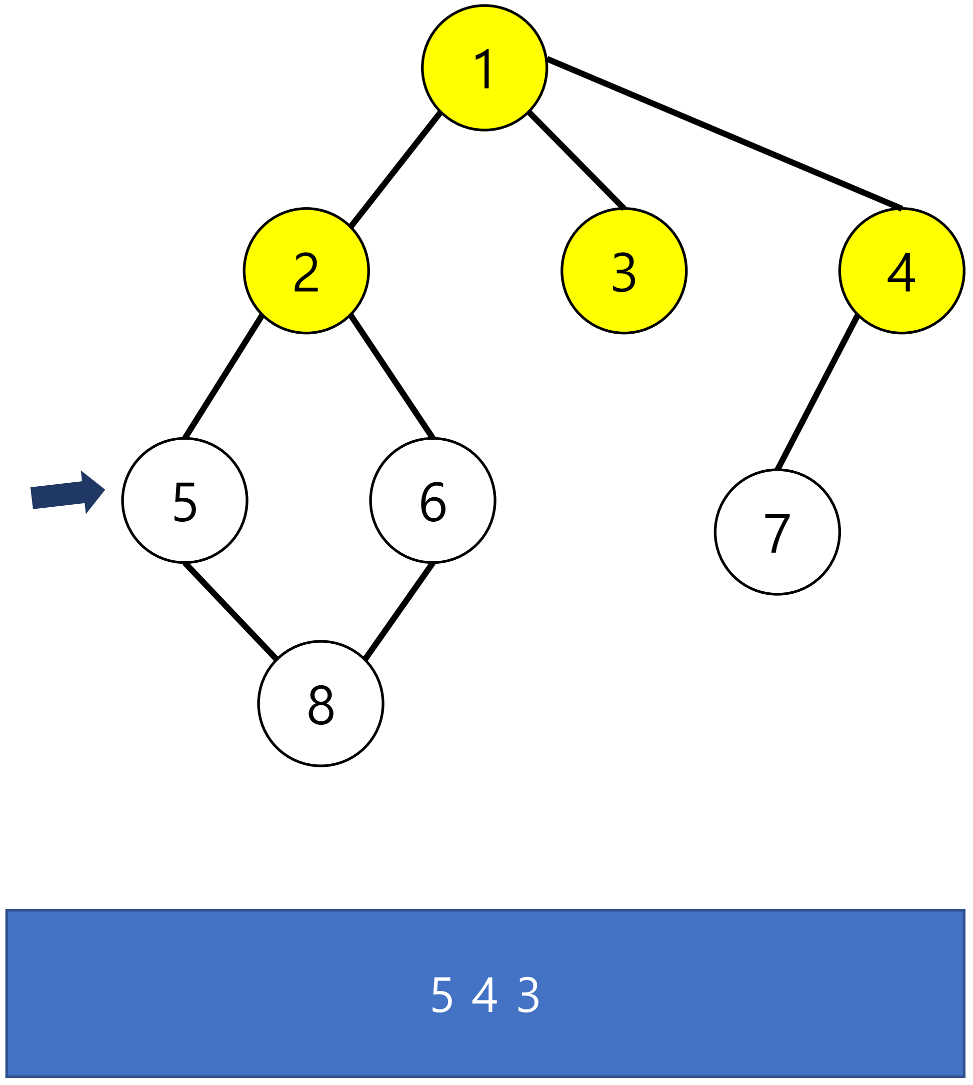
```python
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```

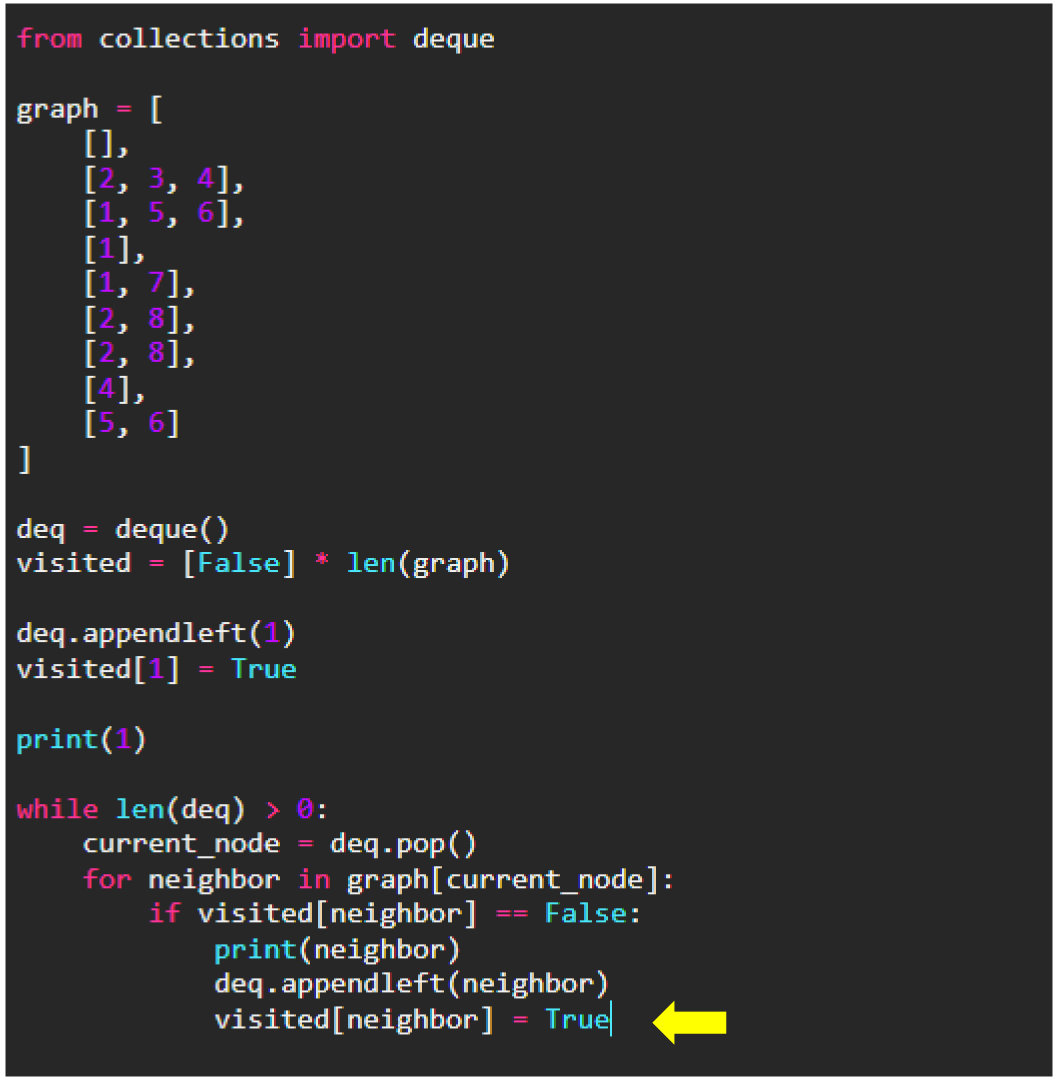


4 3

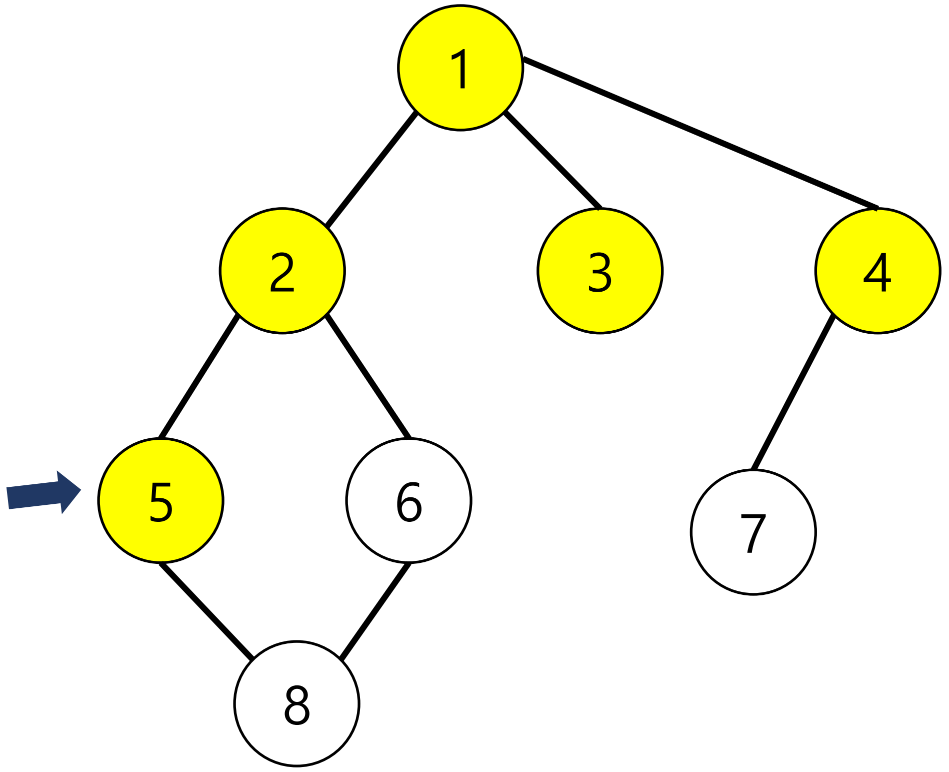


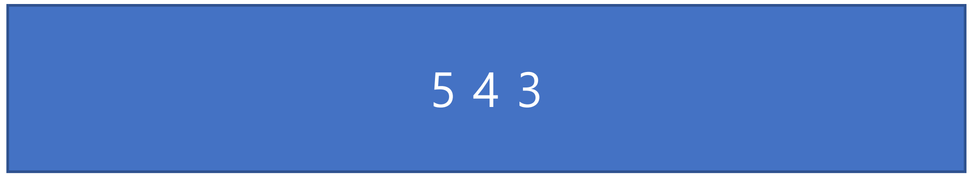
```
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```

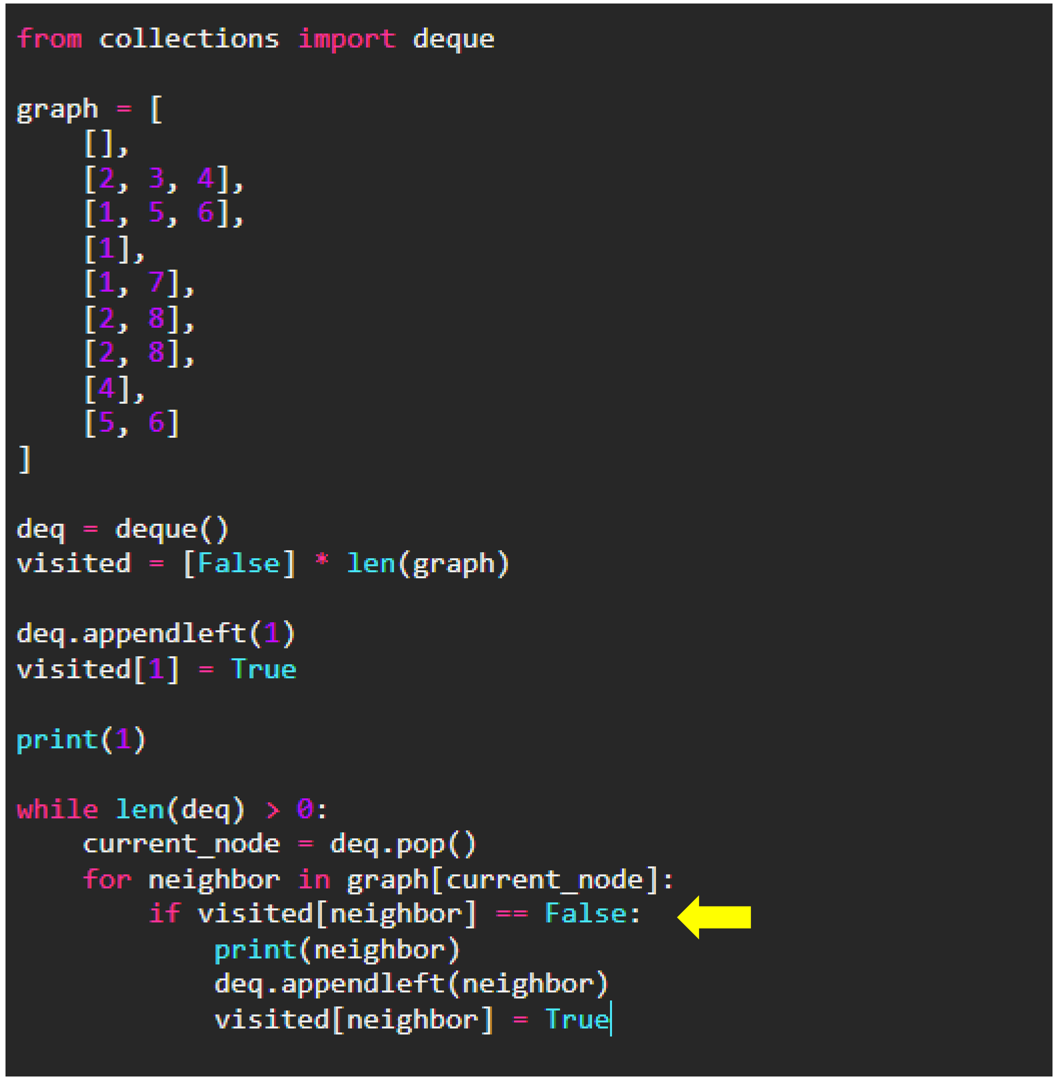




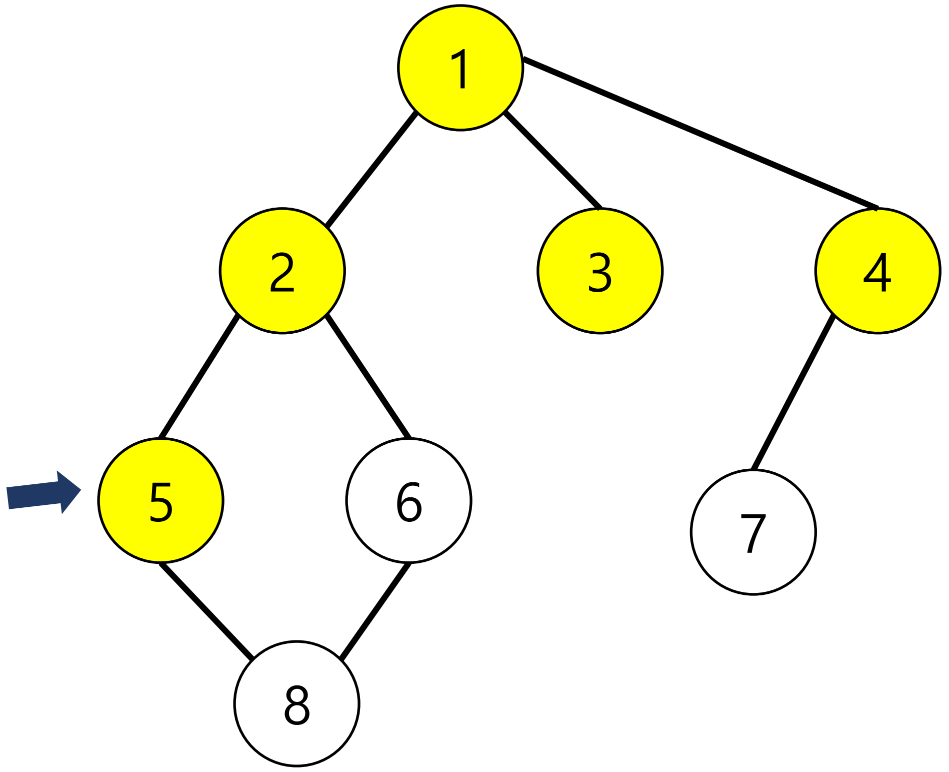
```
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```





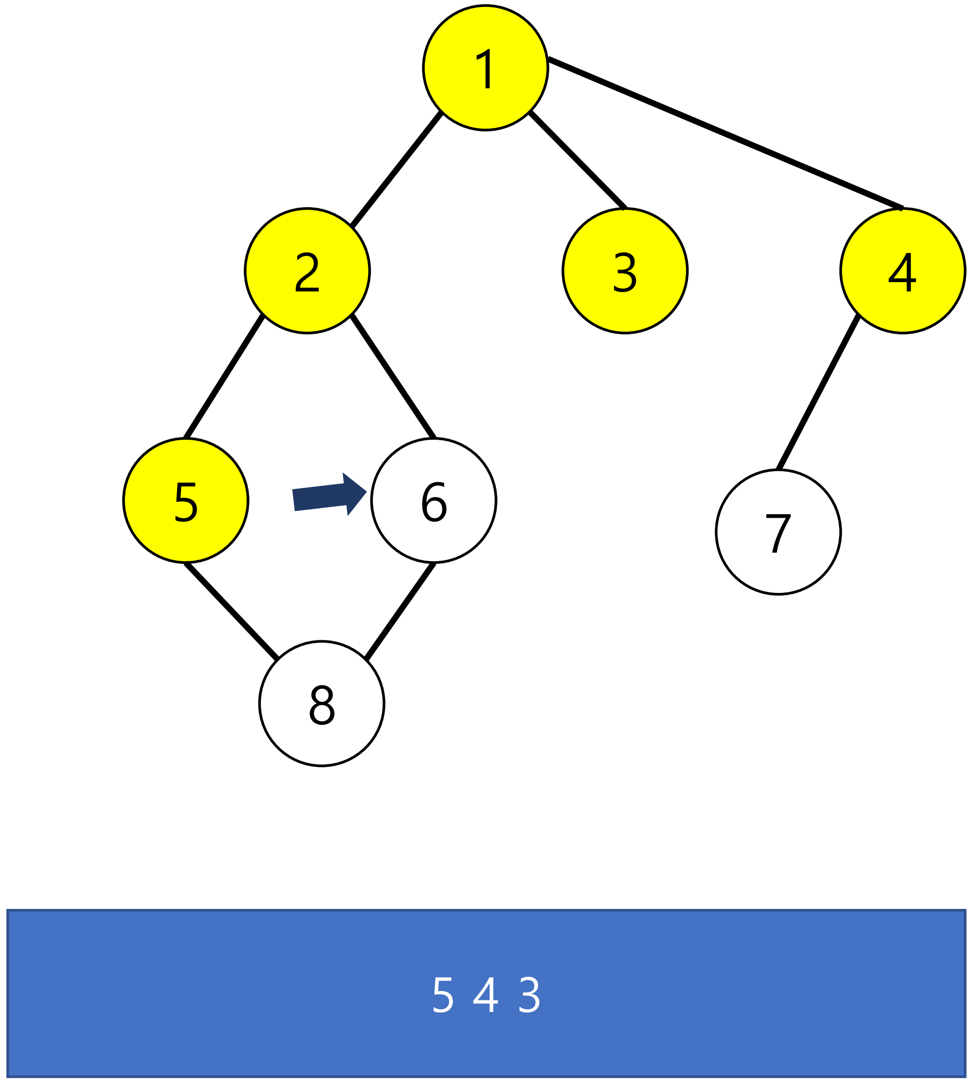


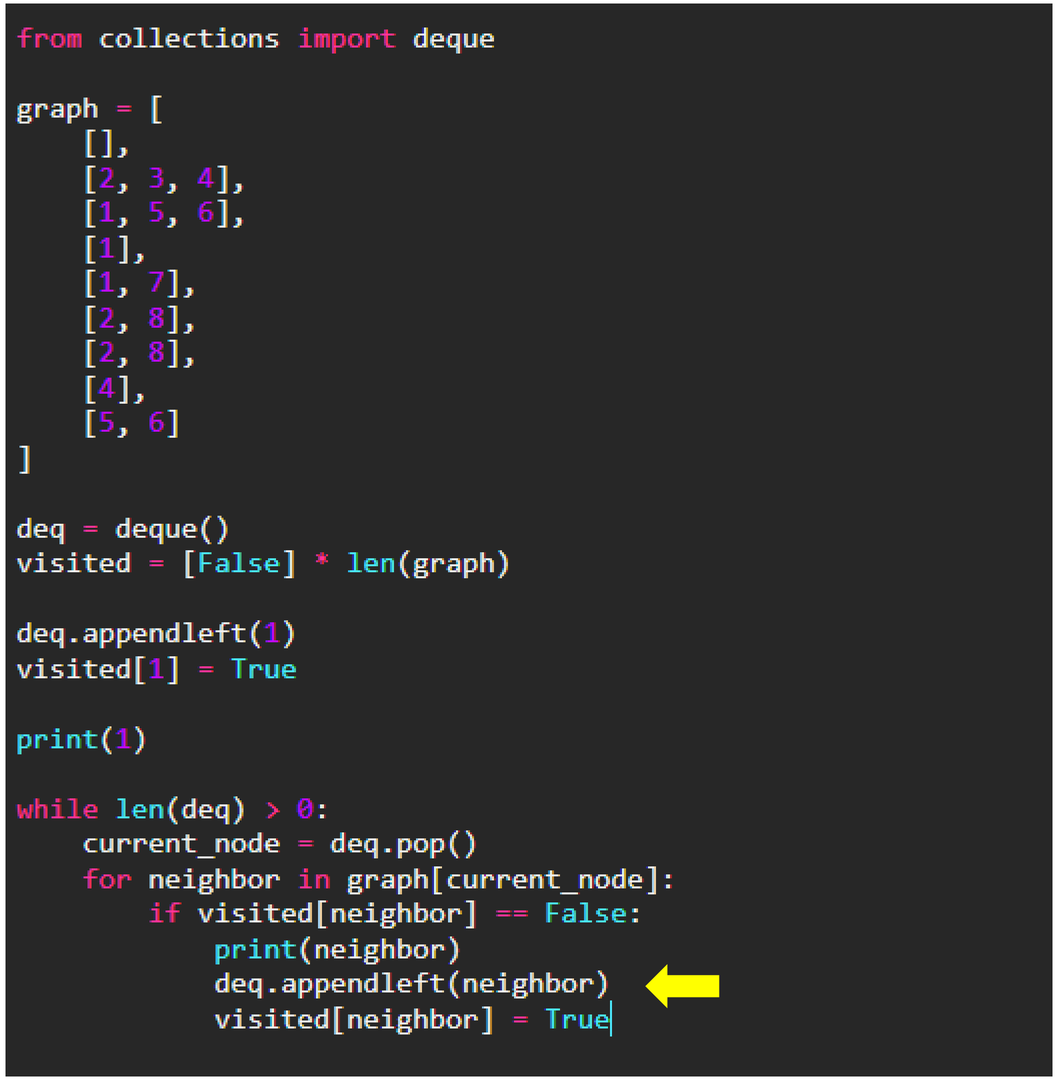
```
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```



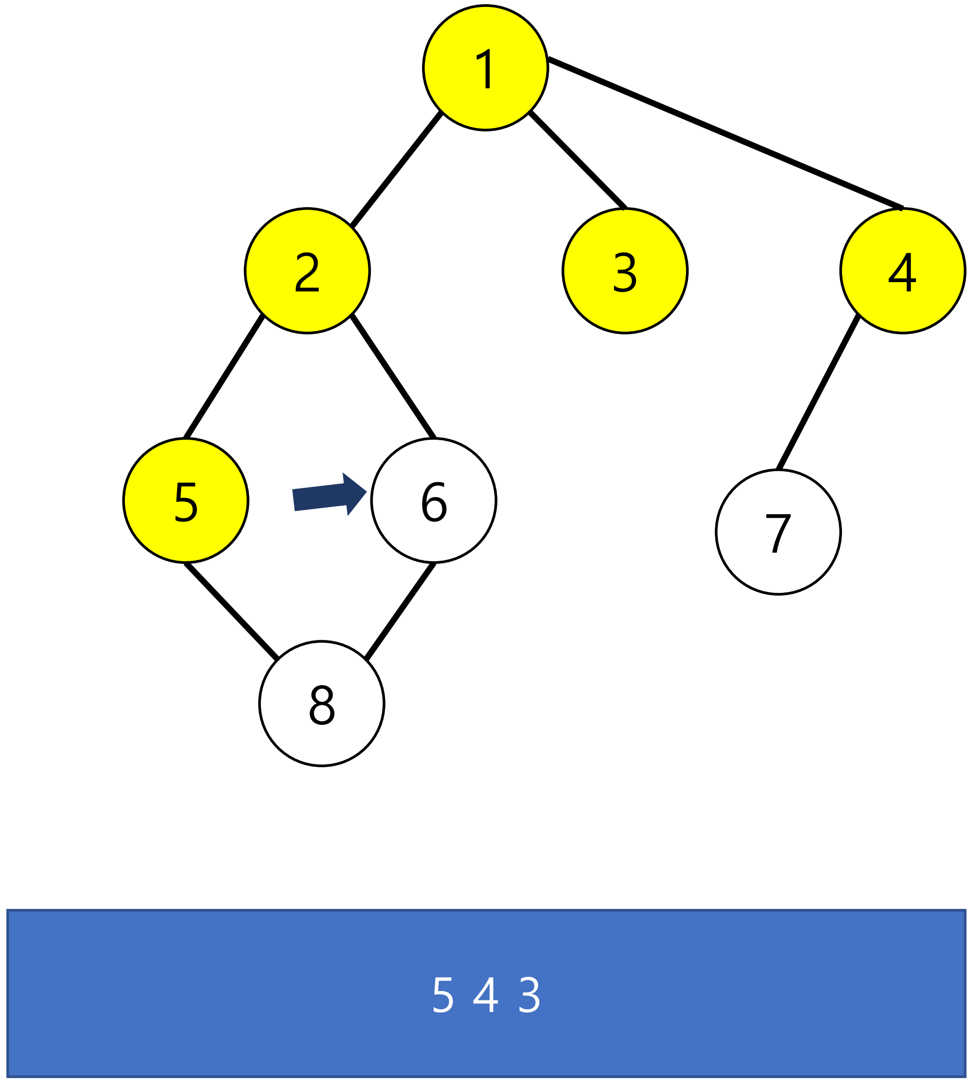
5 4 3

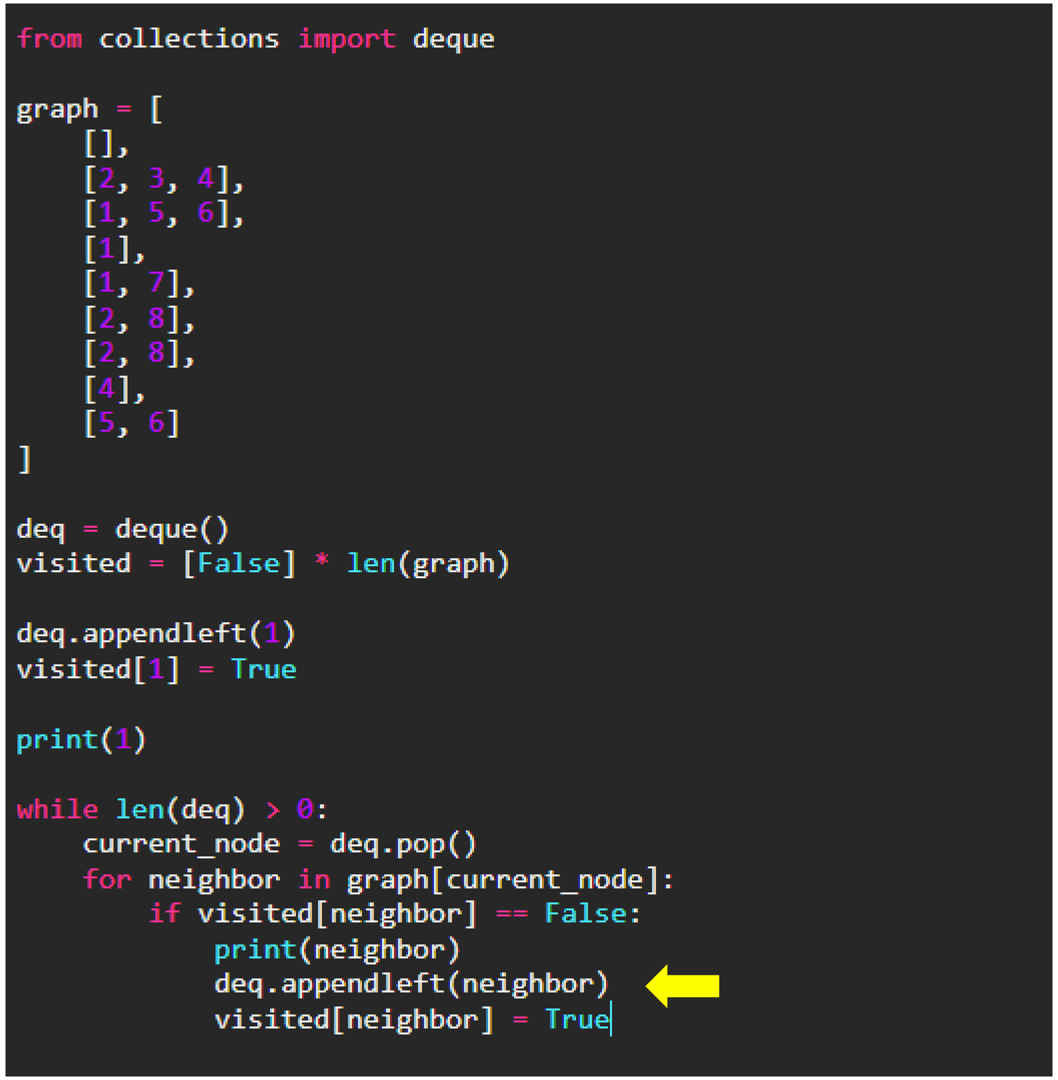
```python
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```



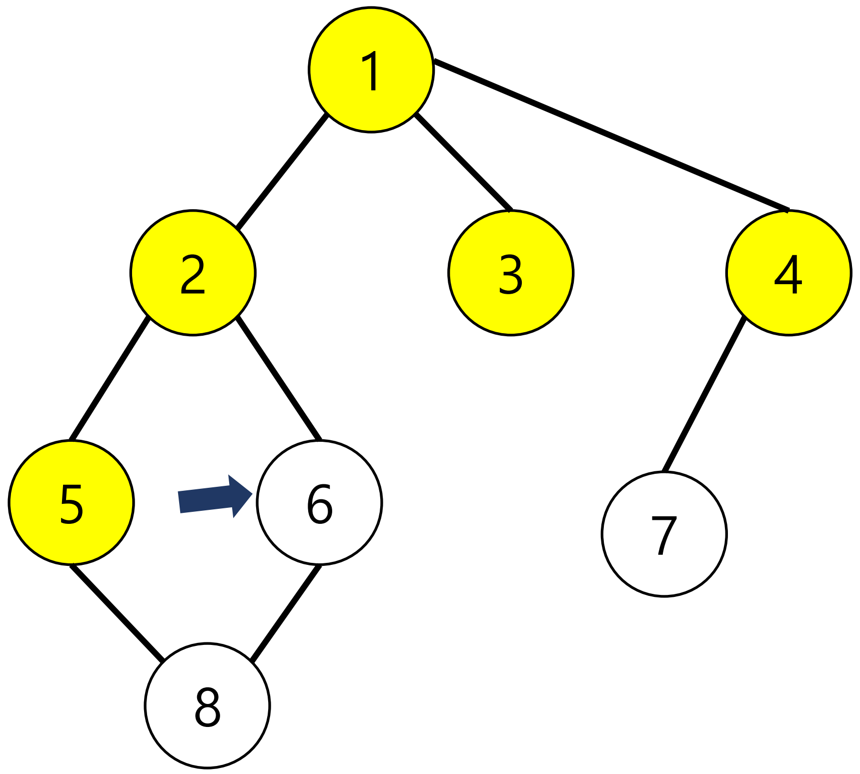


```
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```

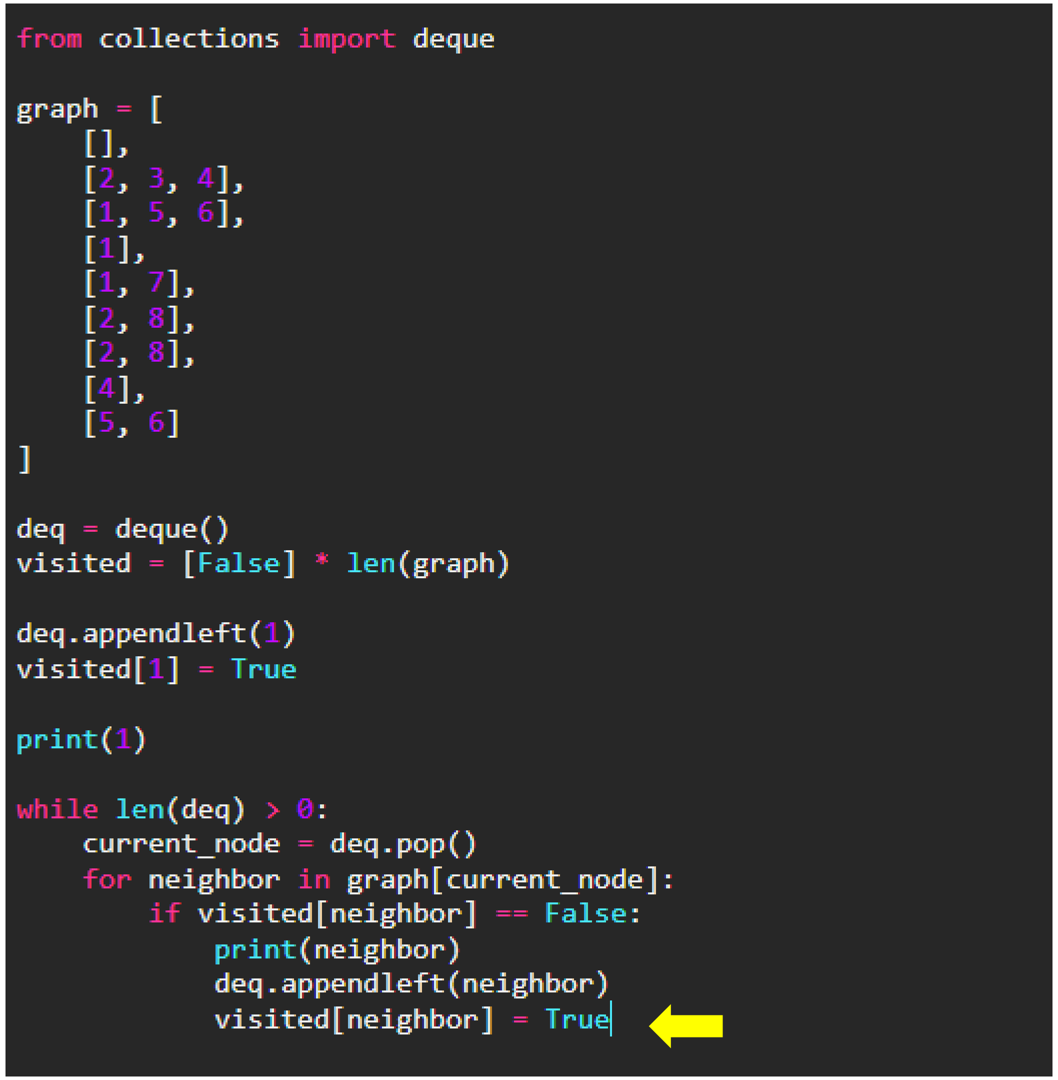




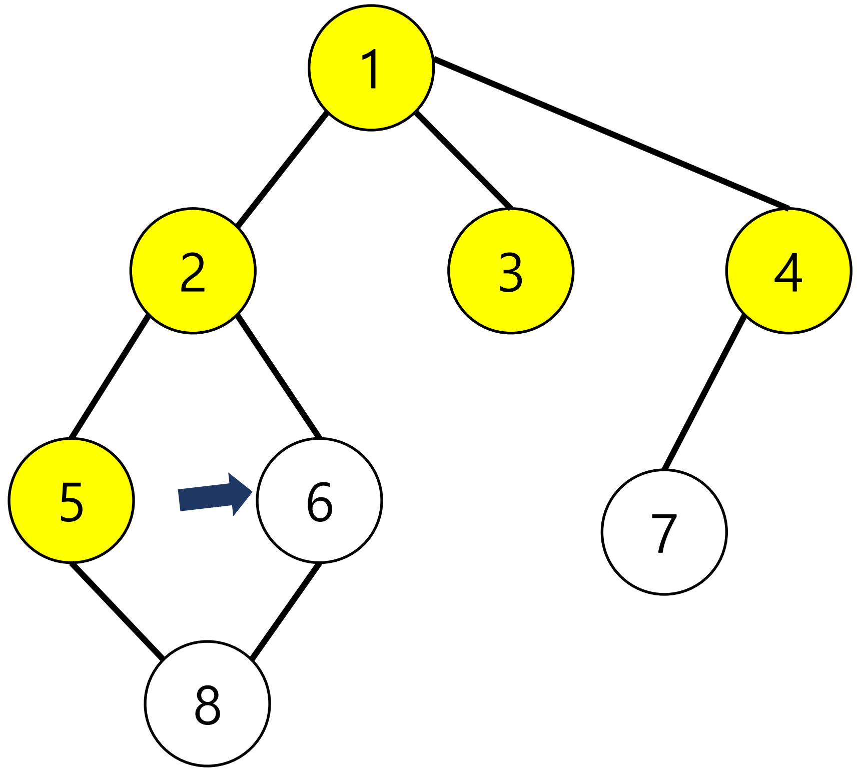
```python
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```



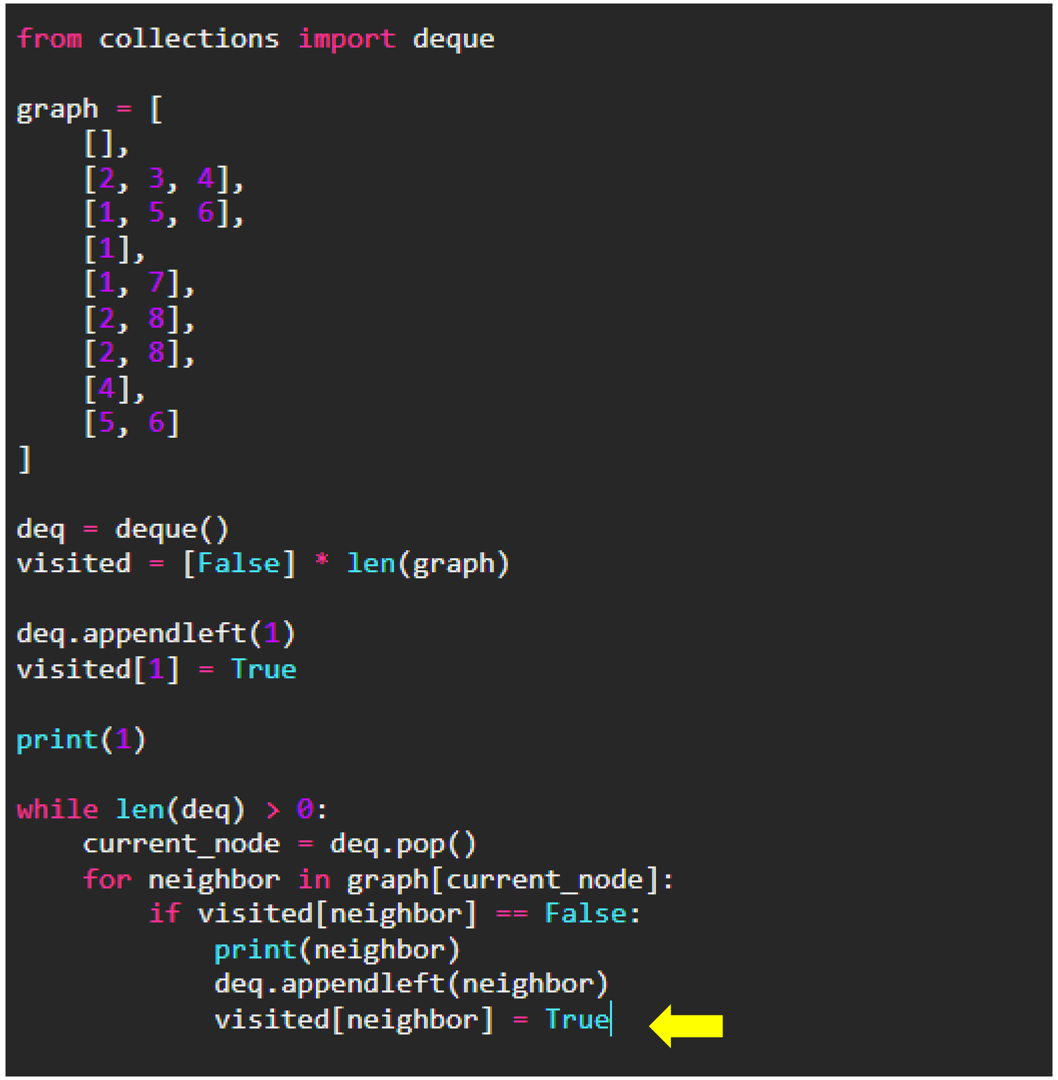
6 5 4 3



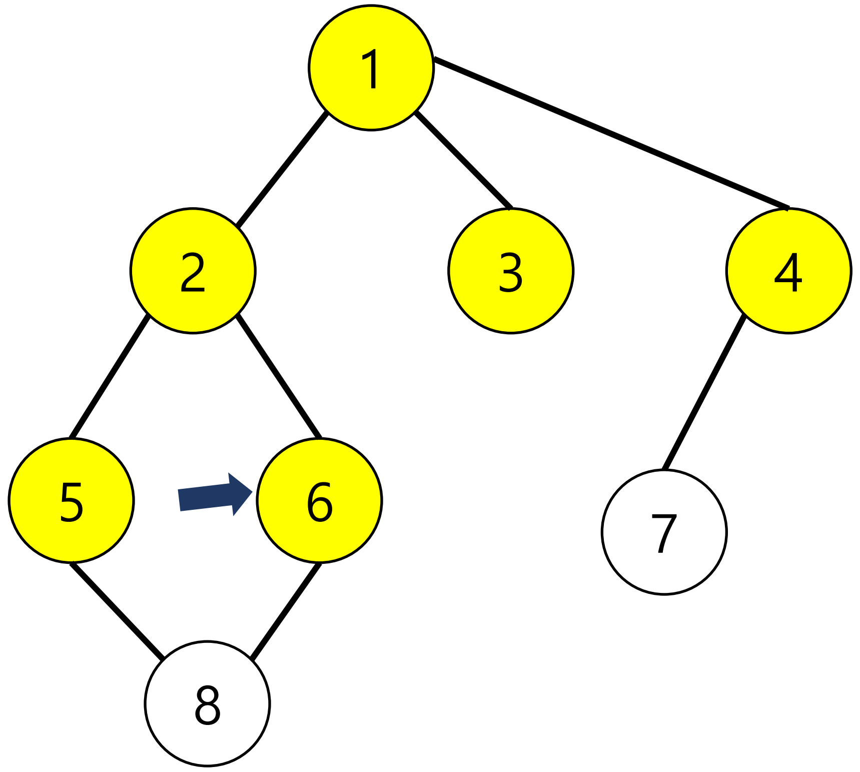
```python
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```



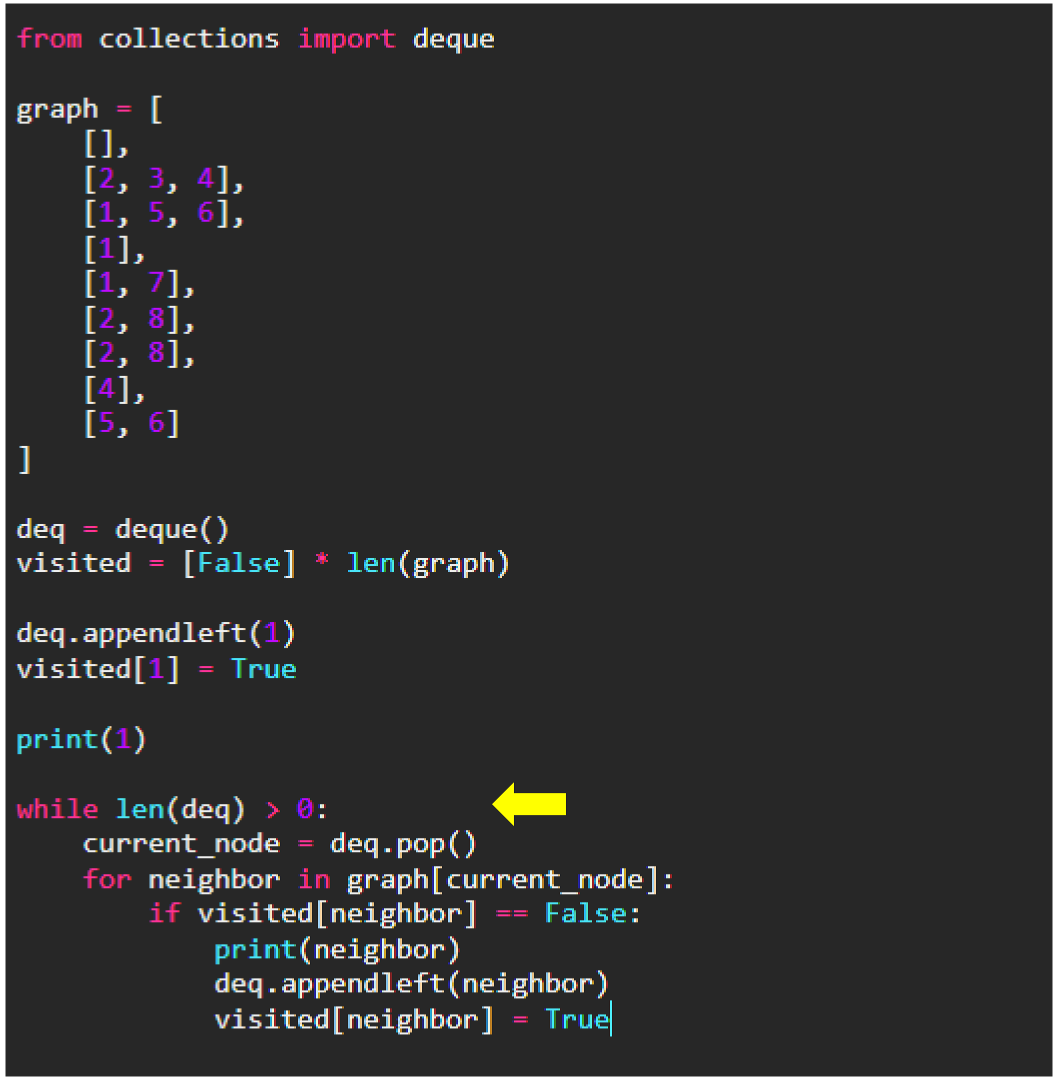
6 5 4 3



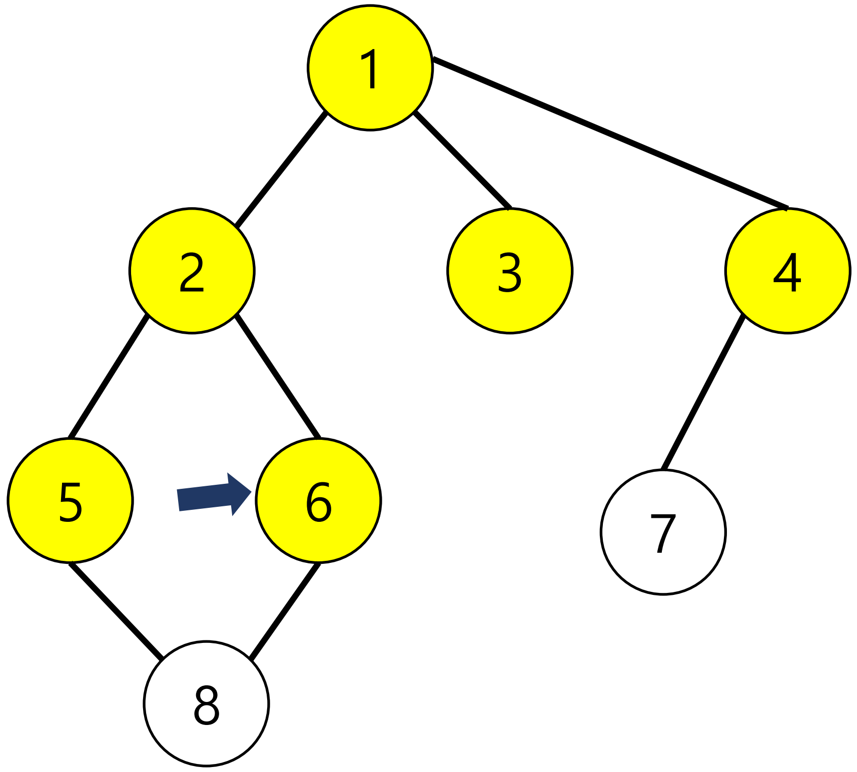
```
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```



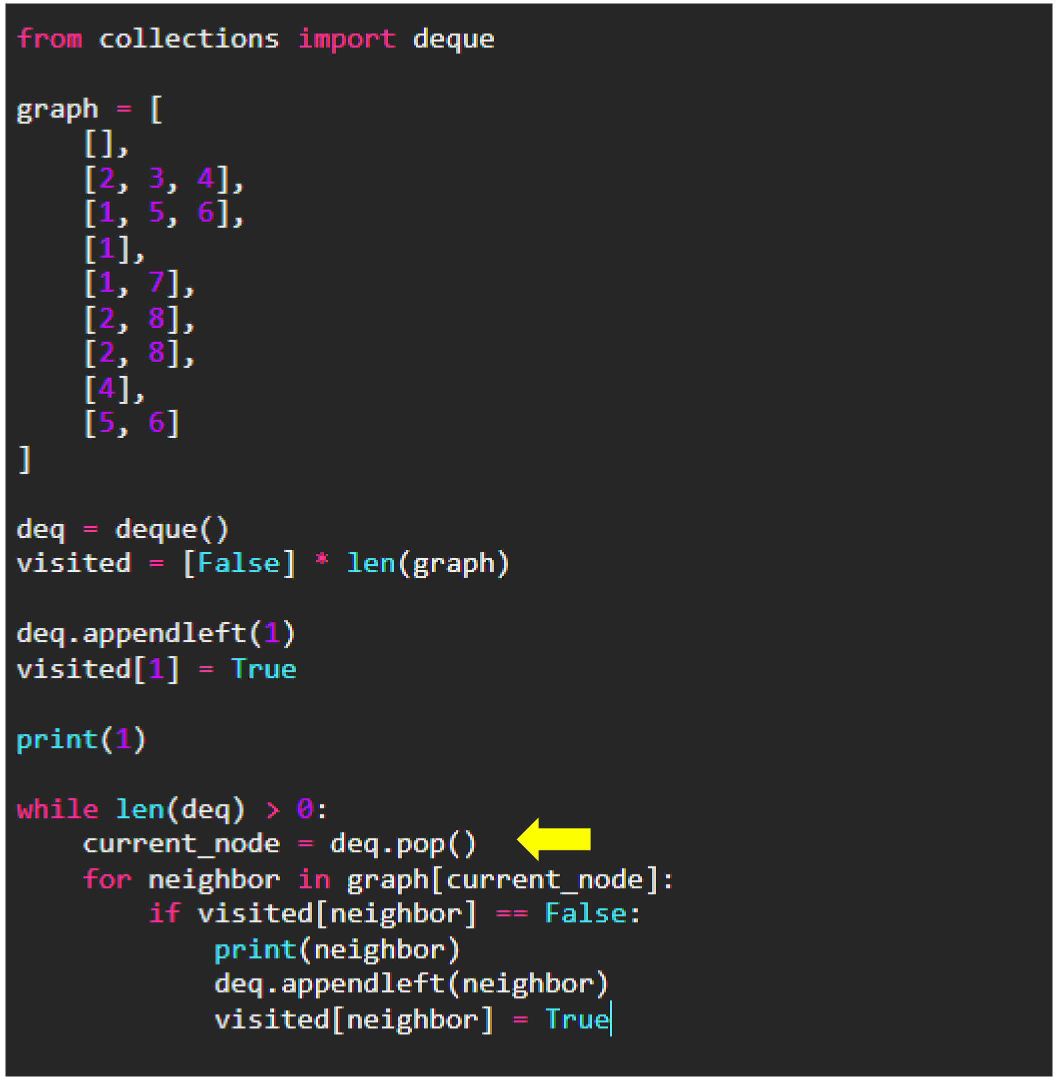
6 5 4 3



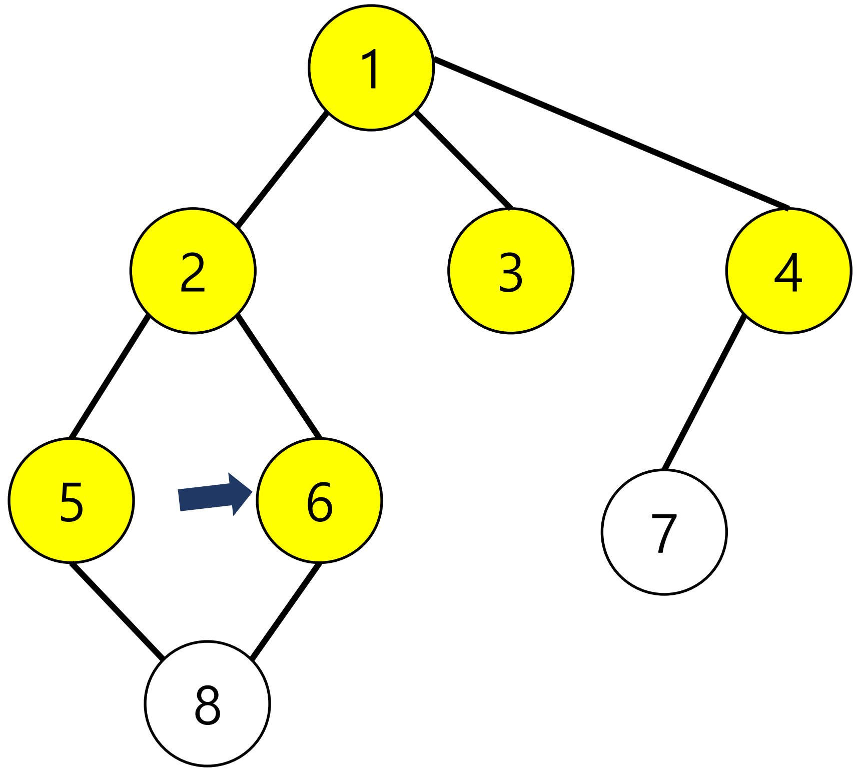
```
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```



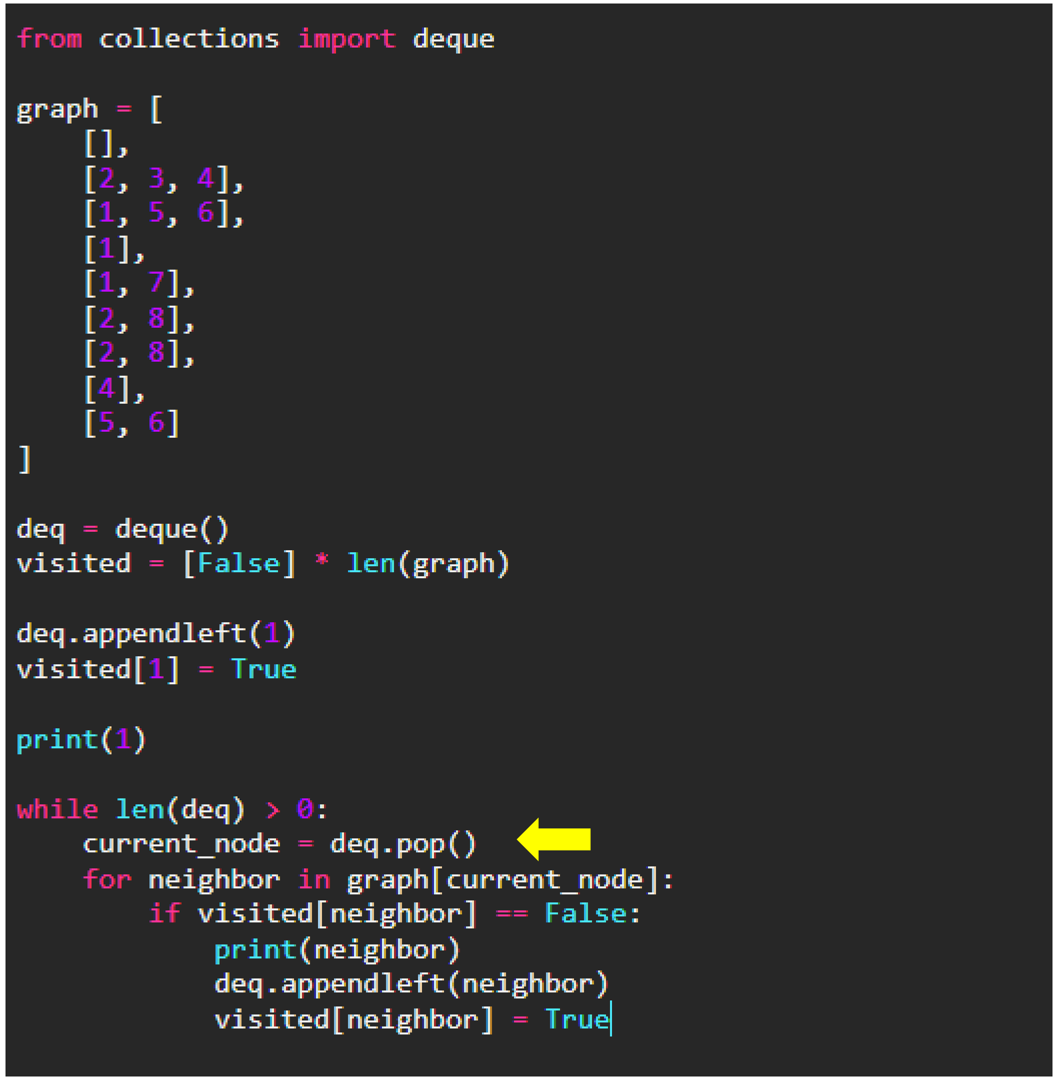
6 5 4 3



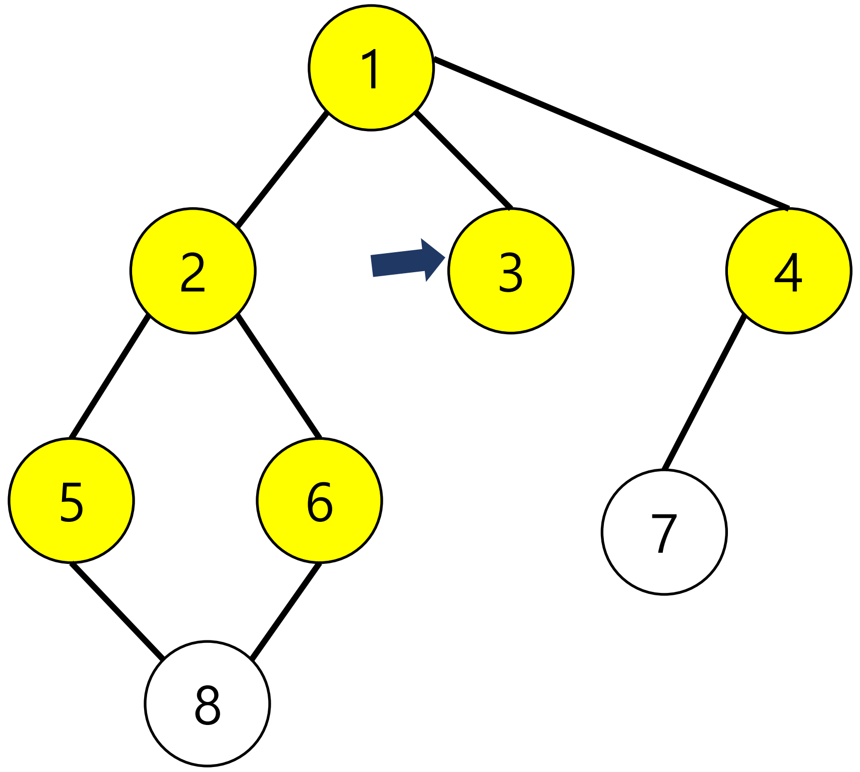
```
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```

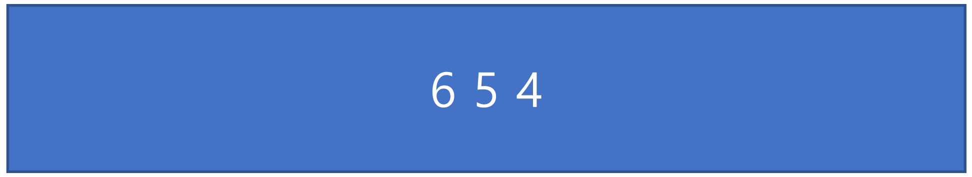


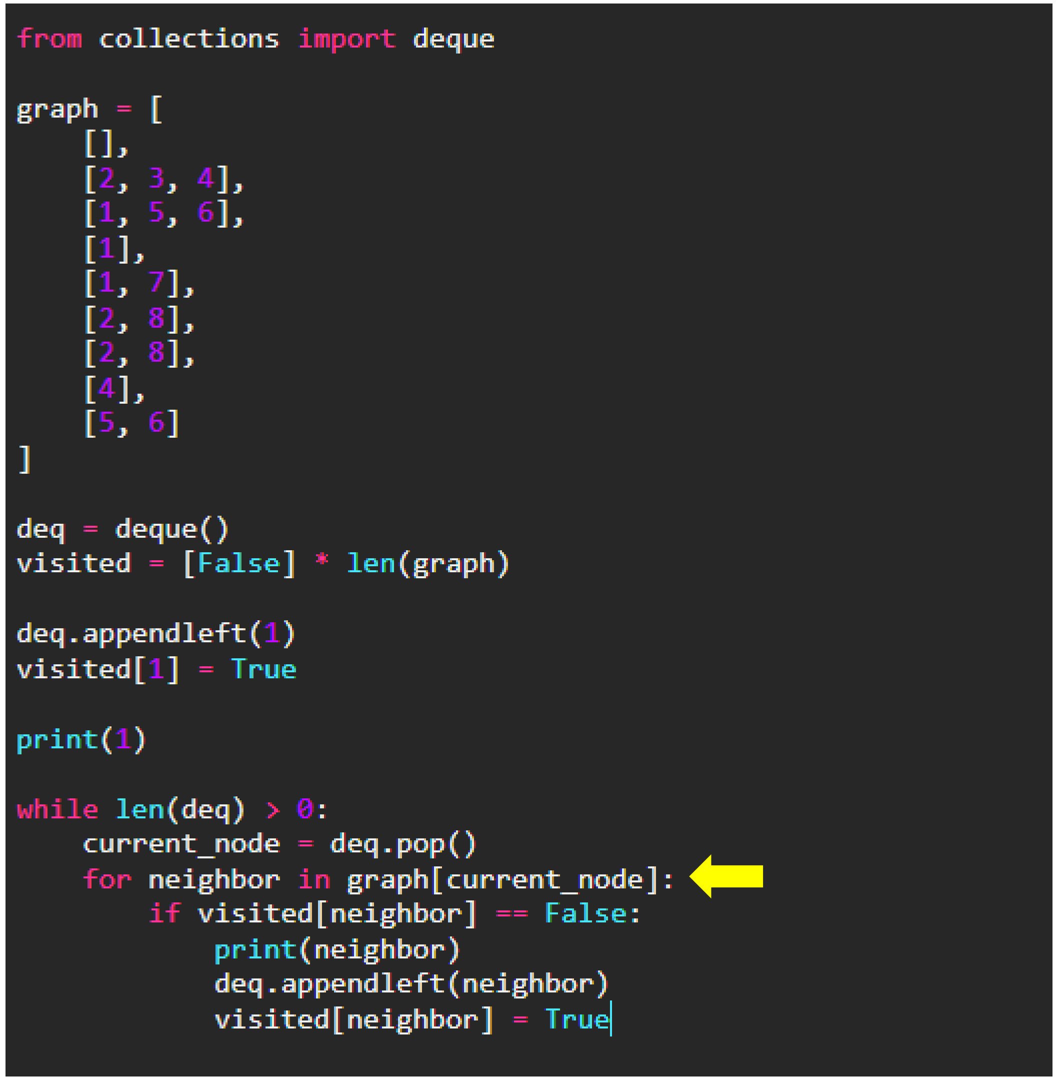
6 5 4 3



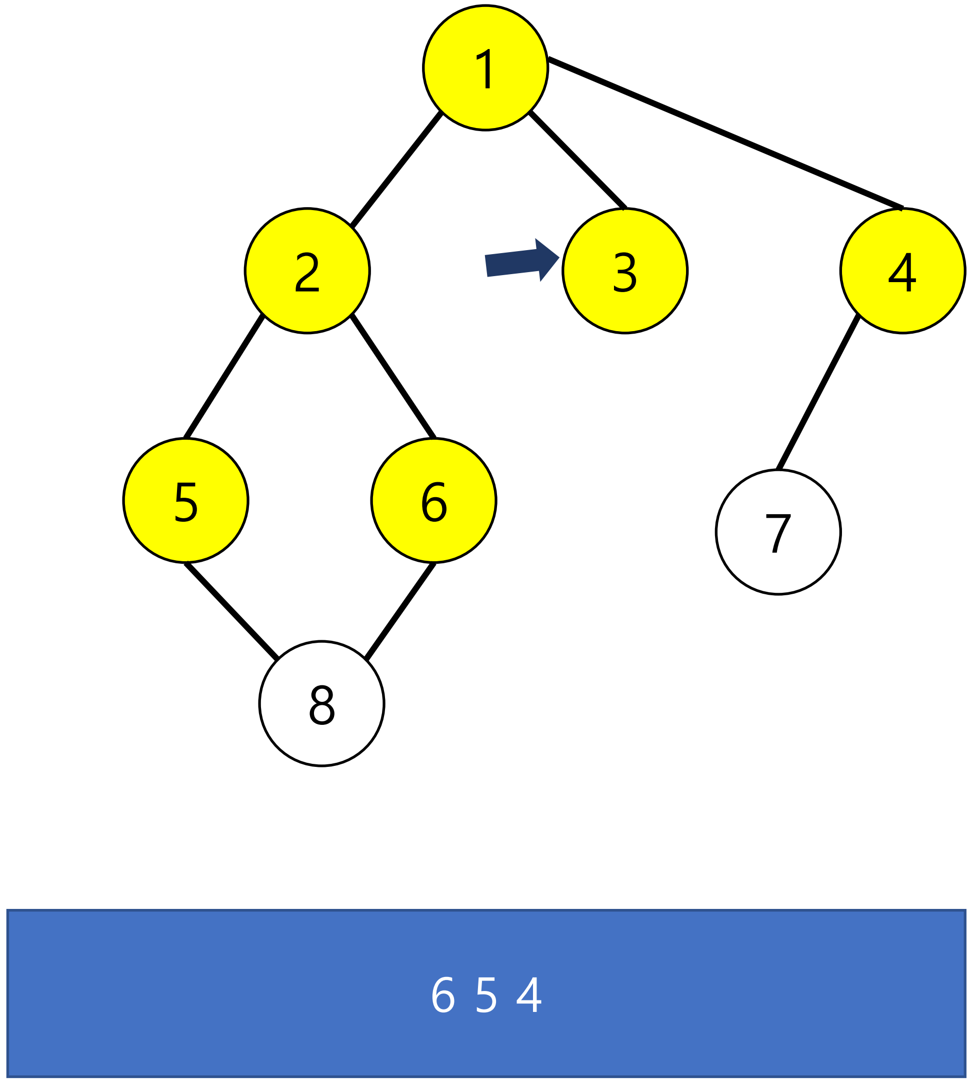
```
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```

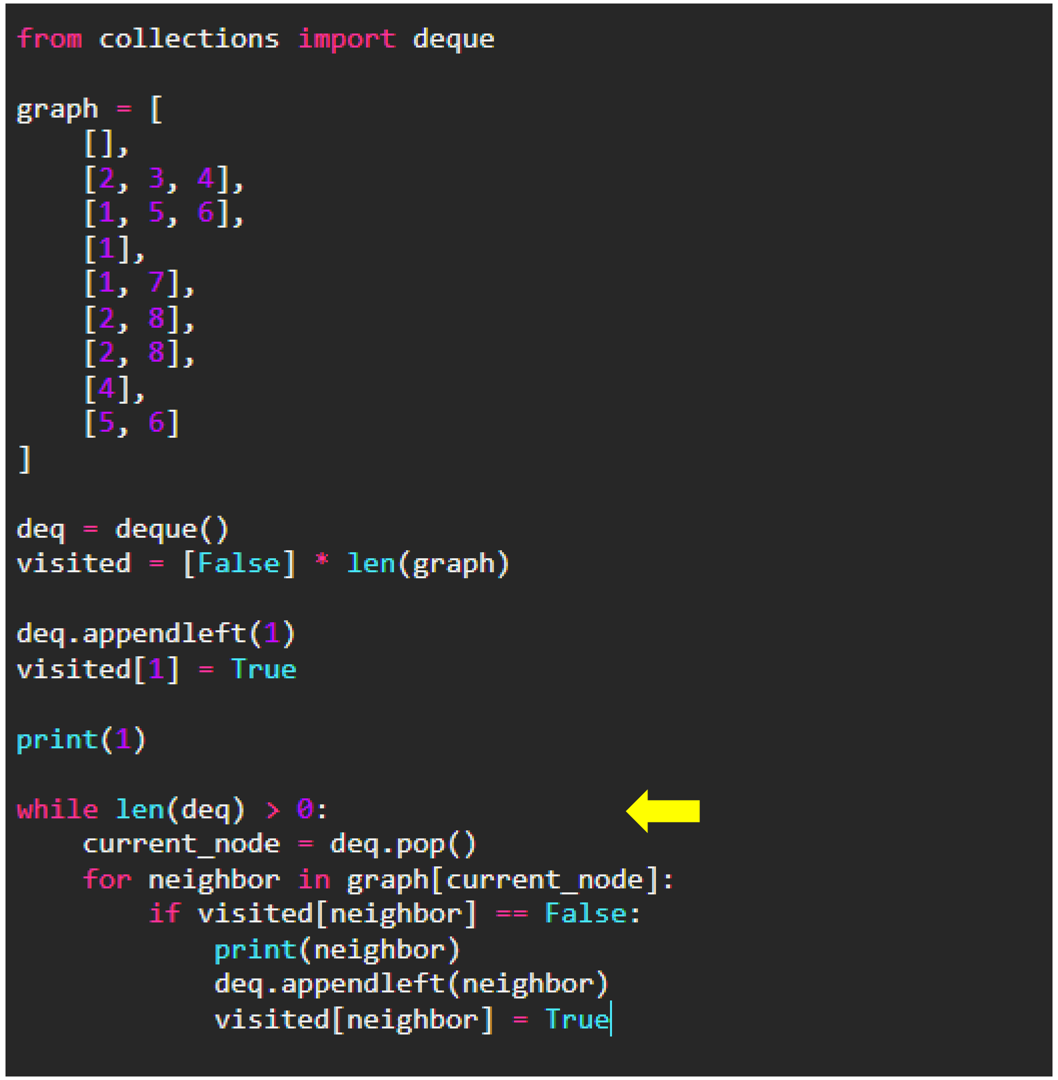




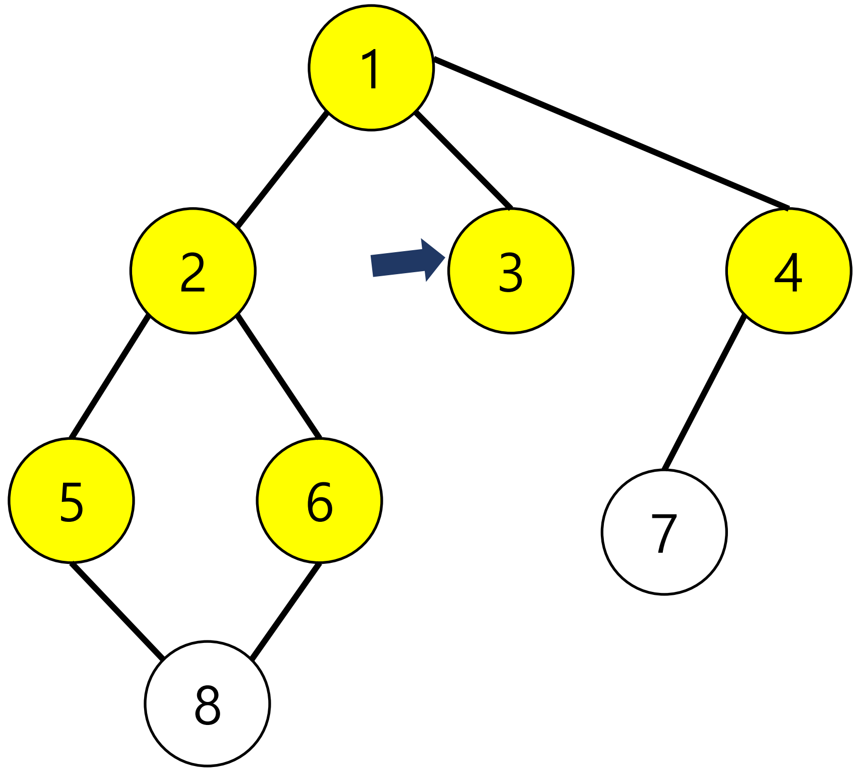


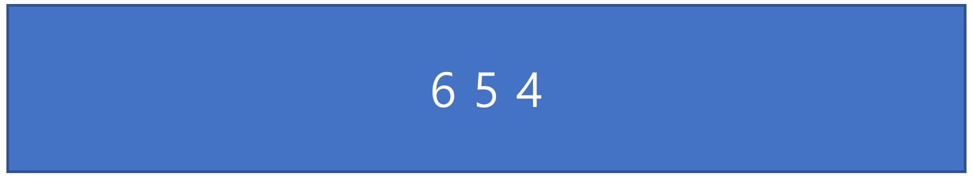
```
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```

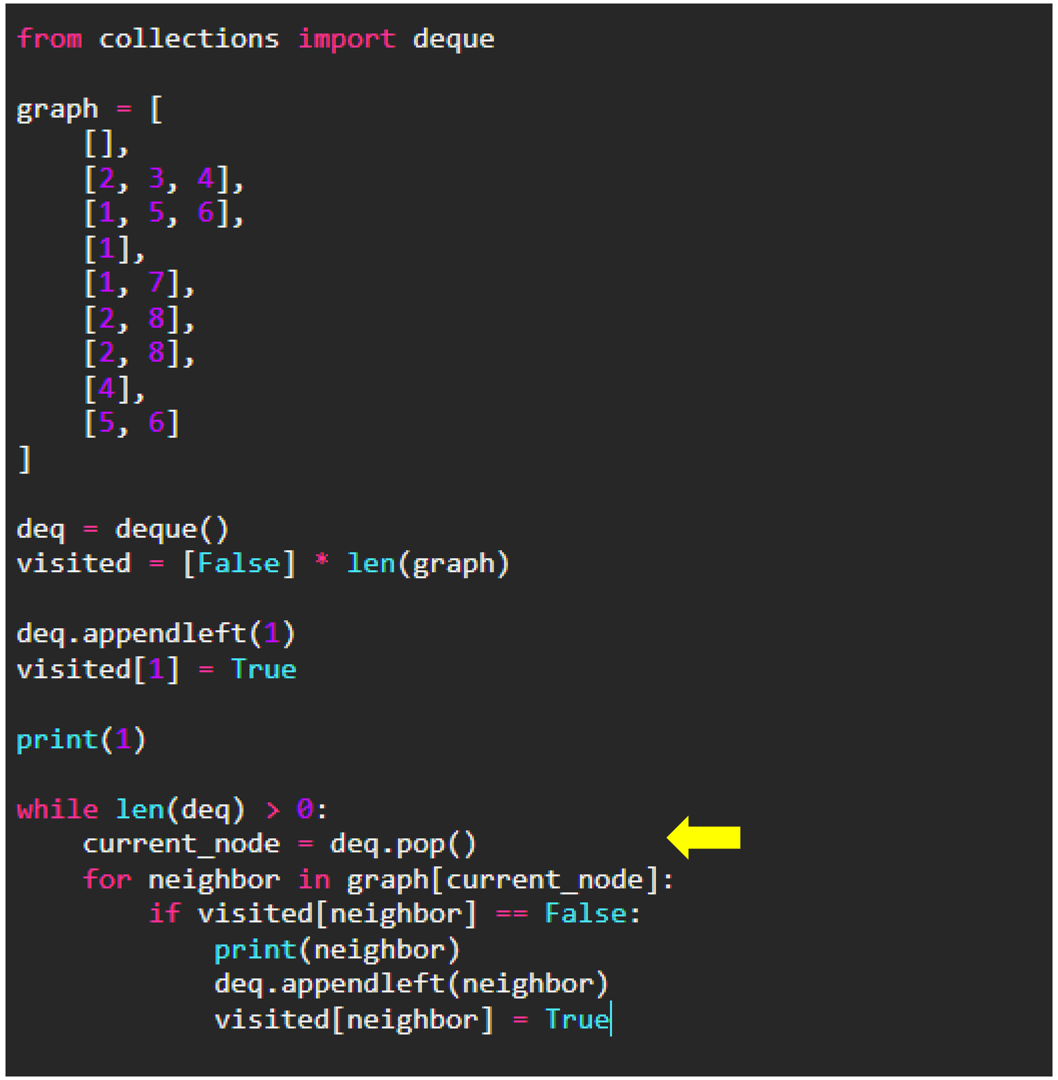




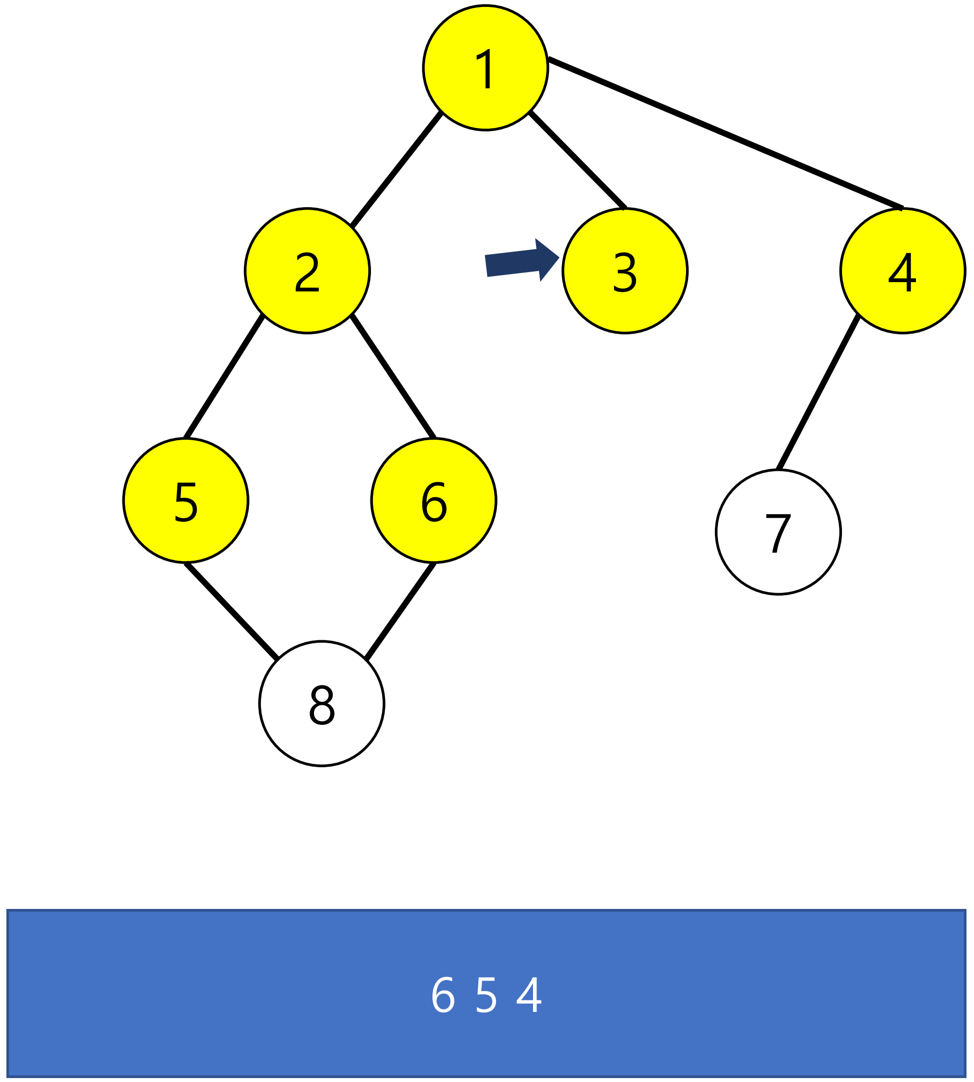
```
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```

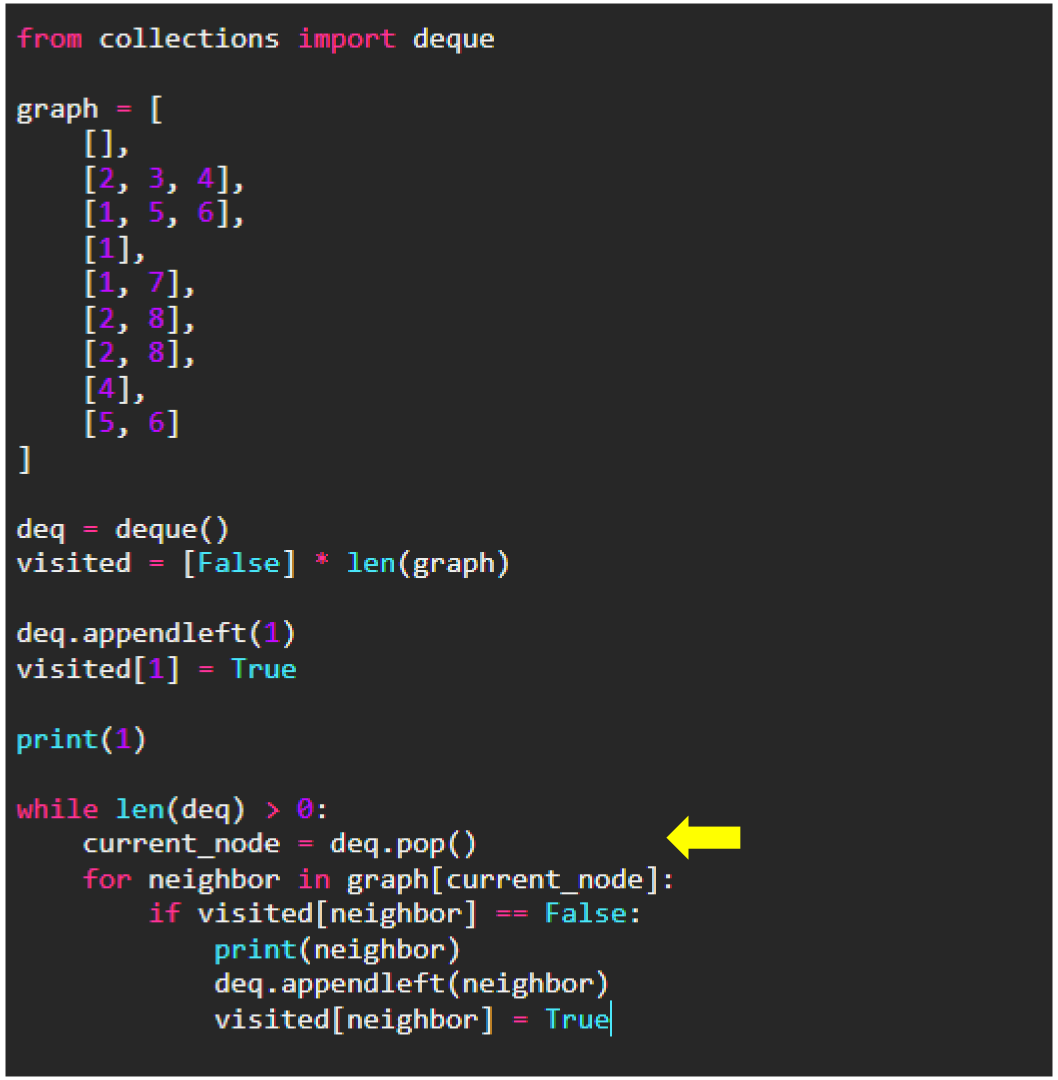




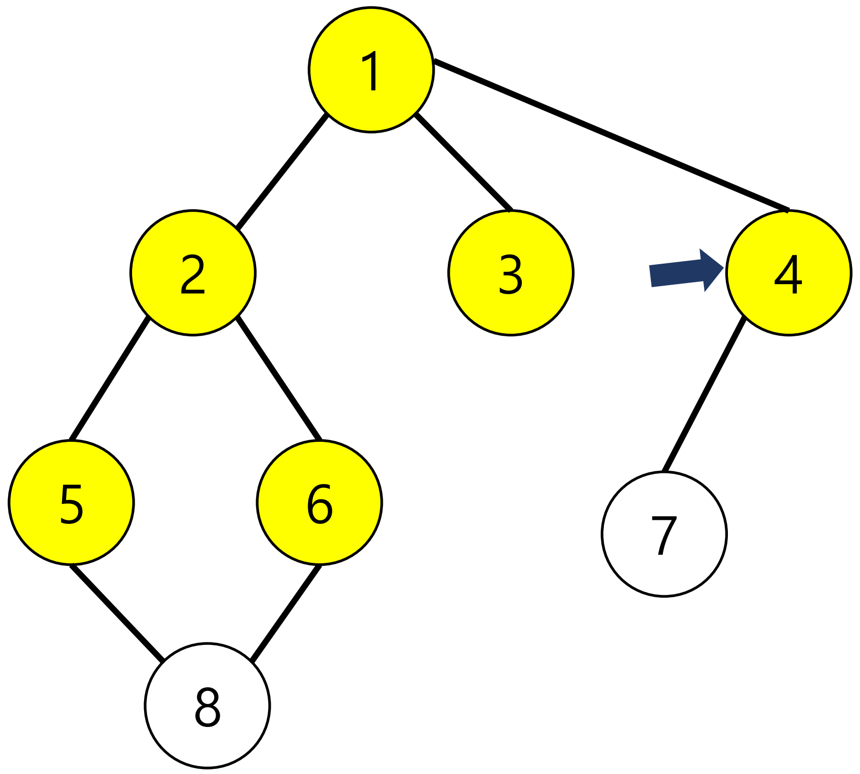


```
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```

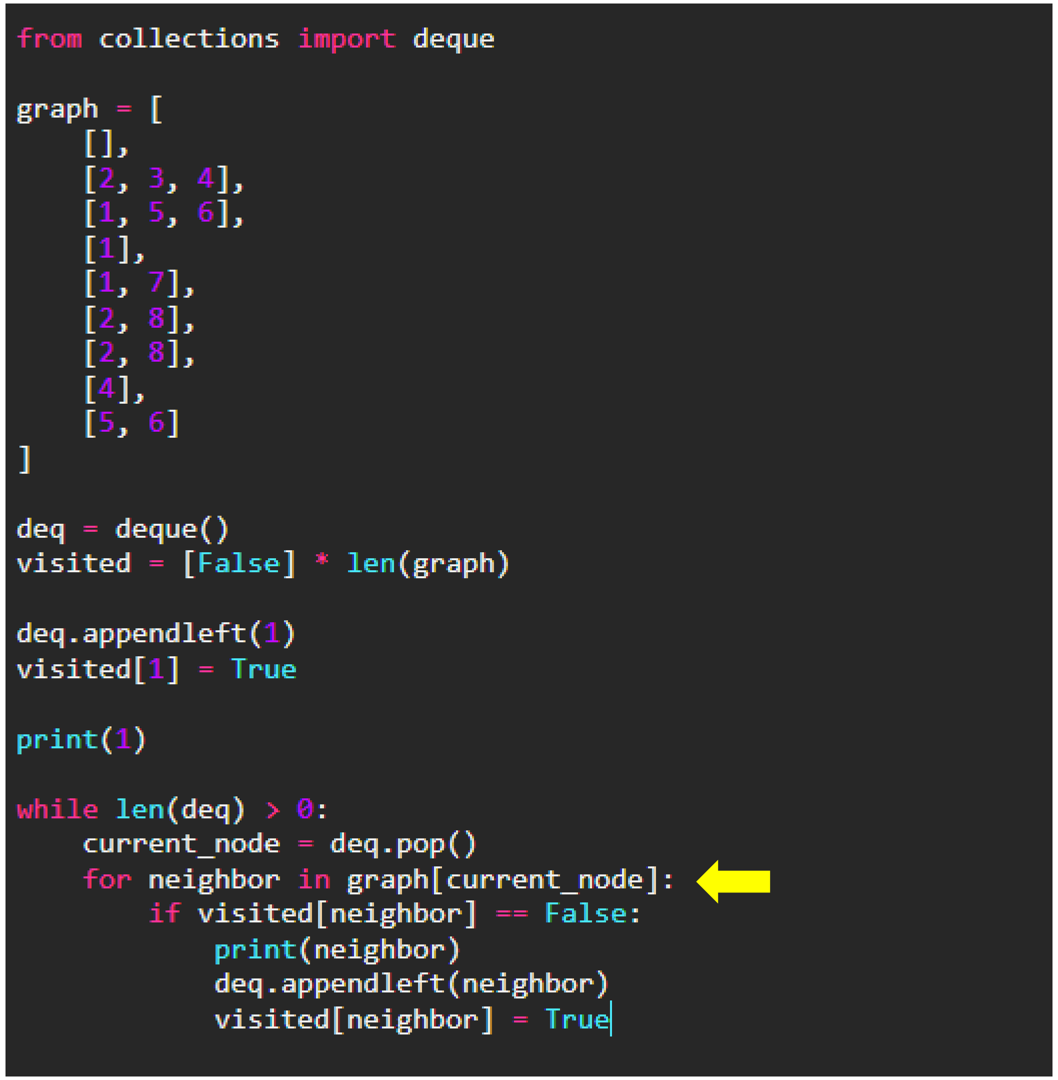


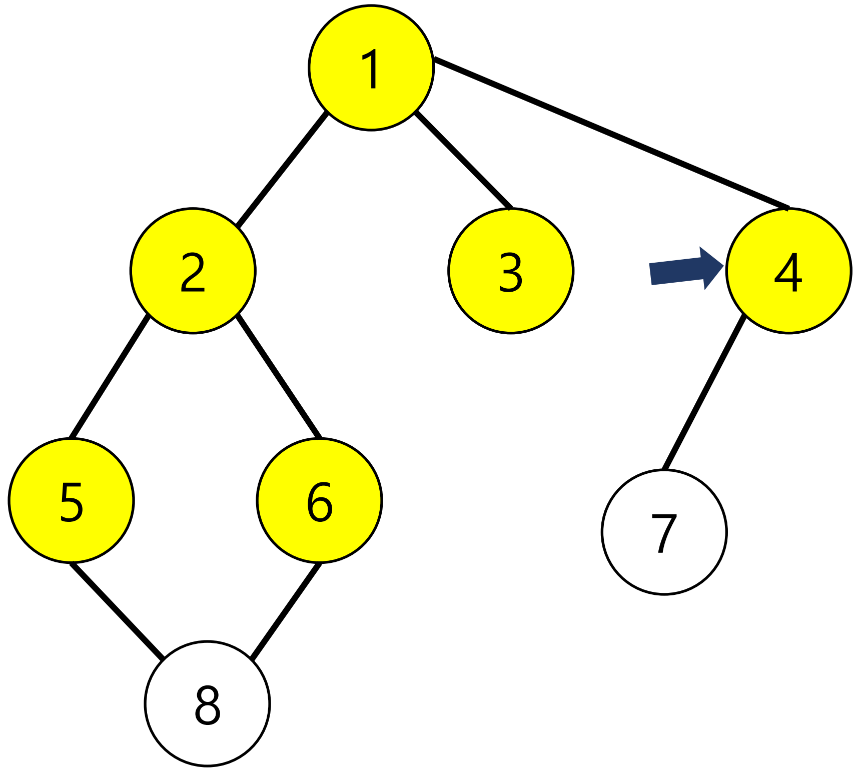


```python
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```



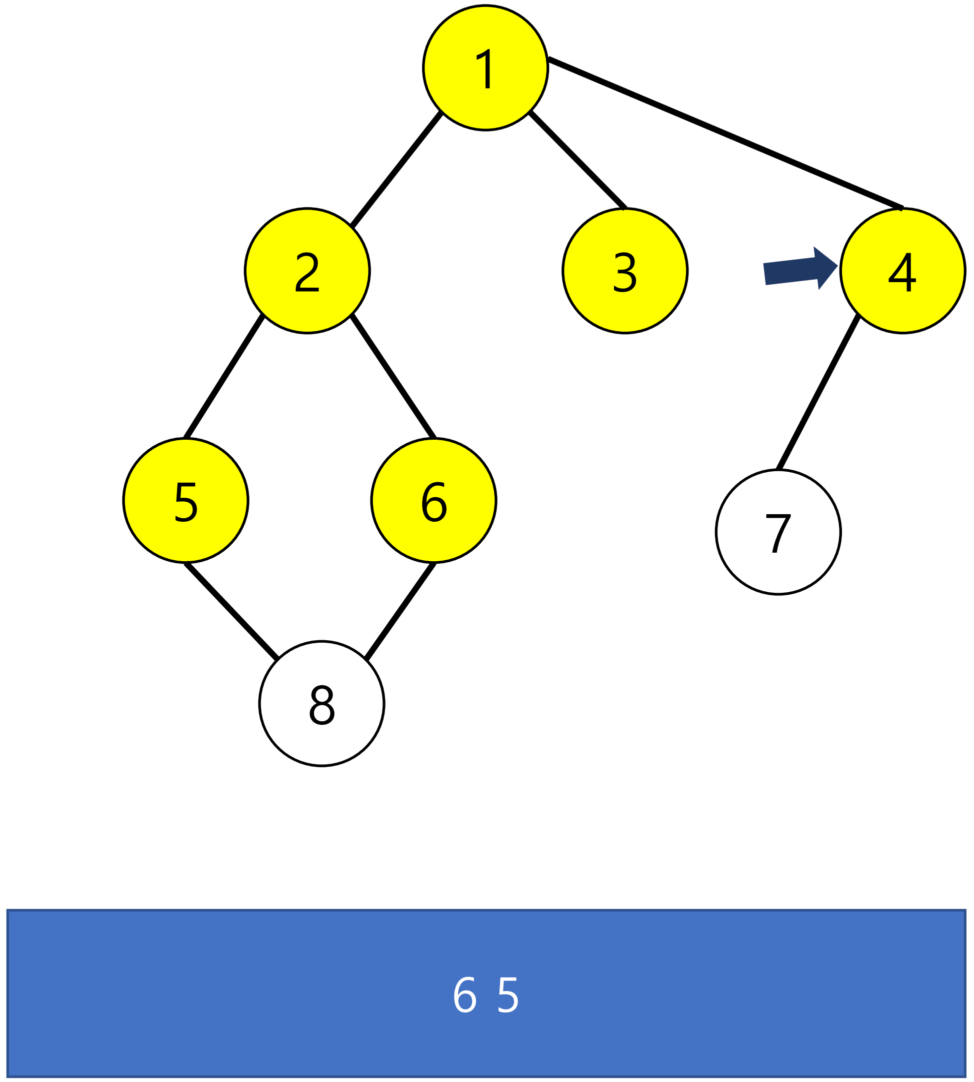
6 5


```
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```

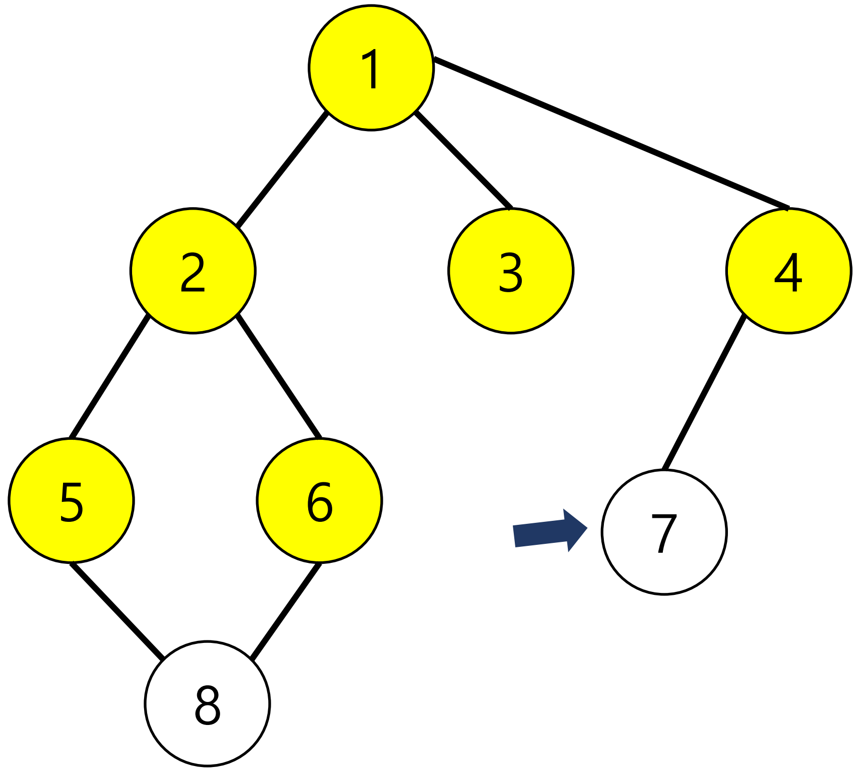


6 5

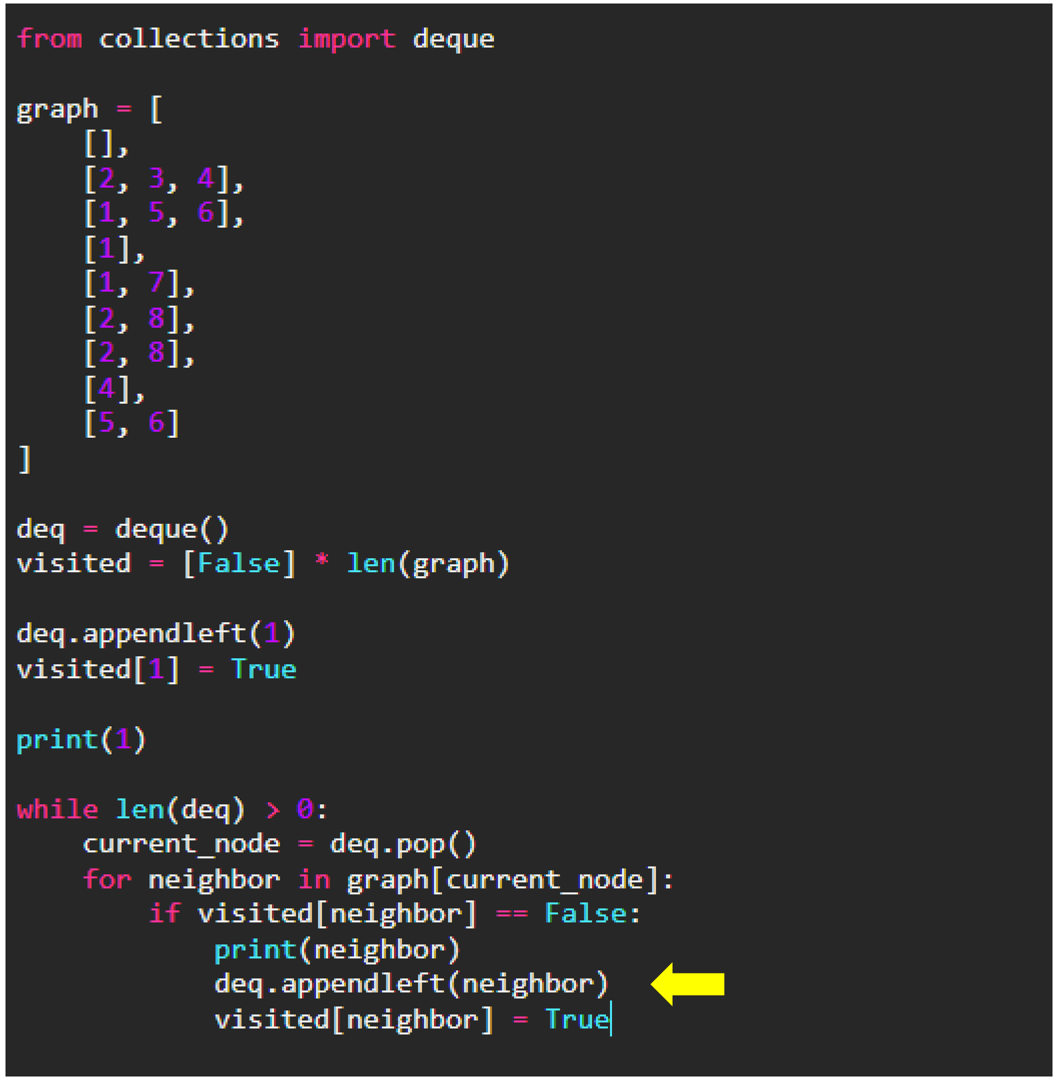
```python
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```



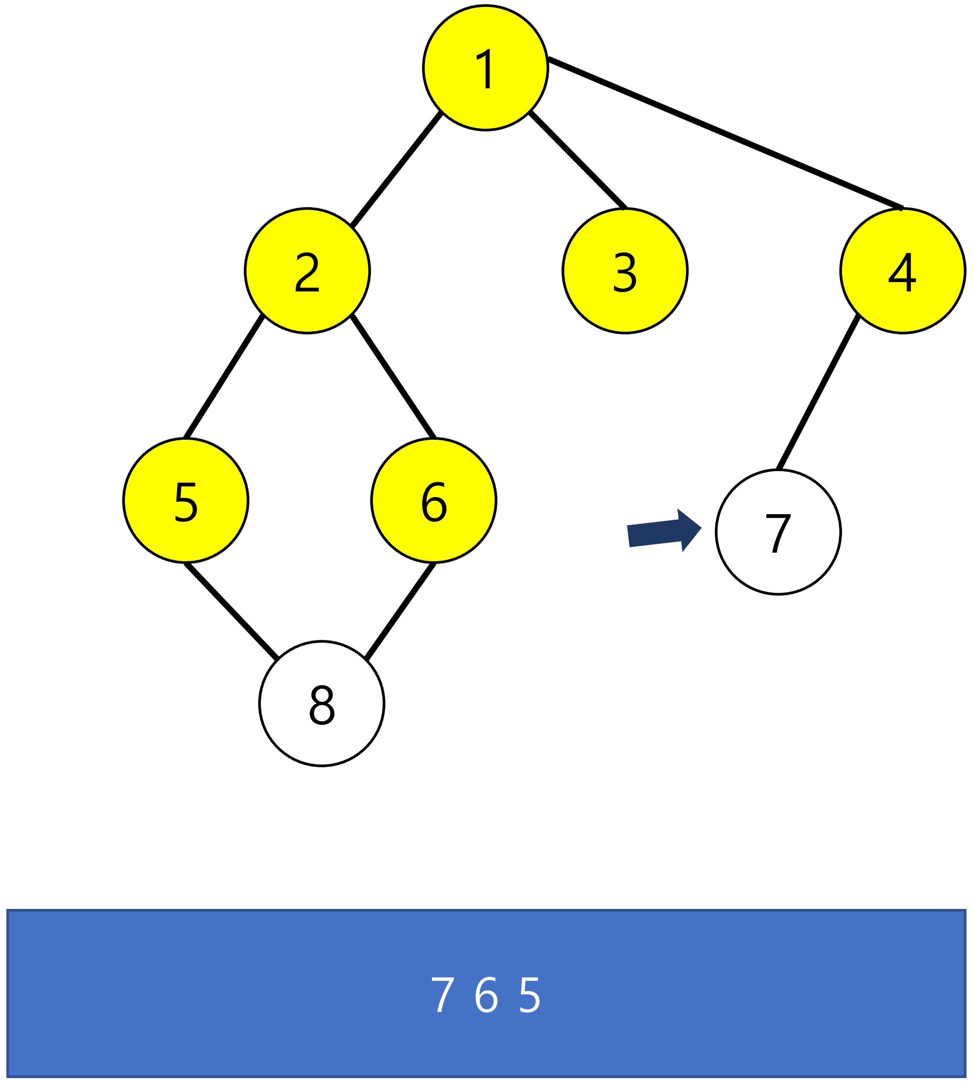
```python
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```

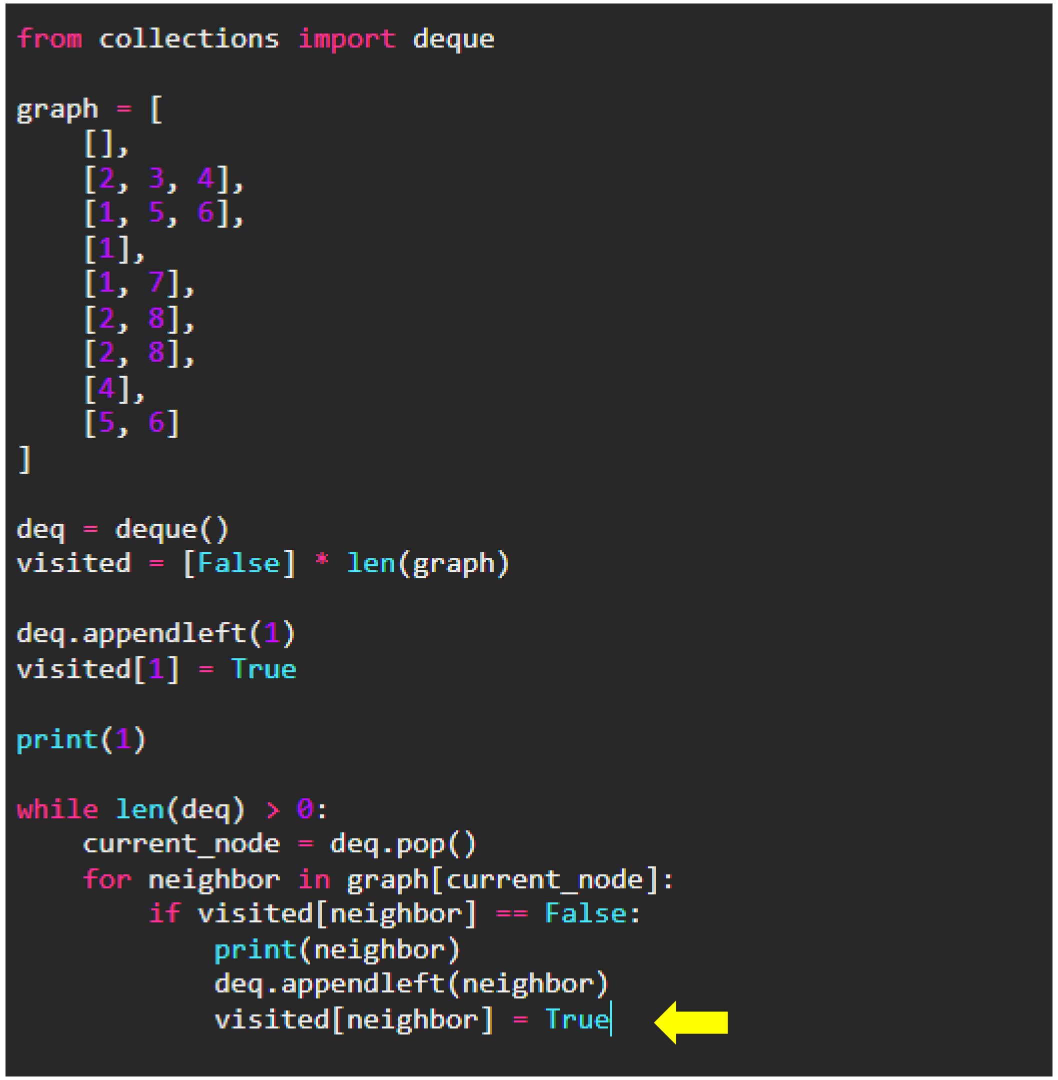


6 5

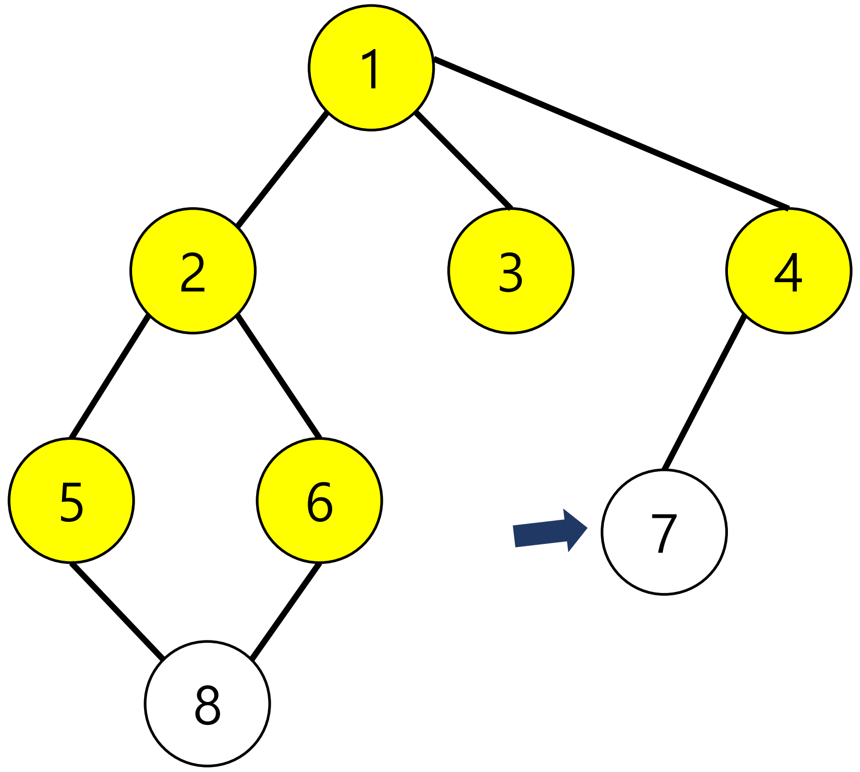


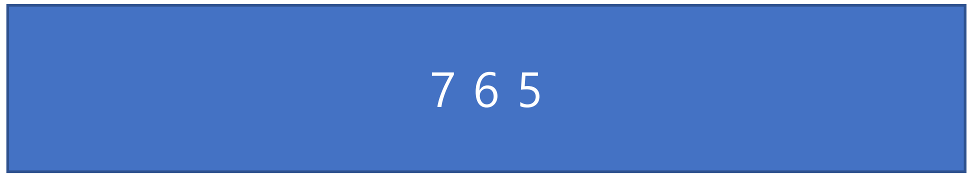
```
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```

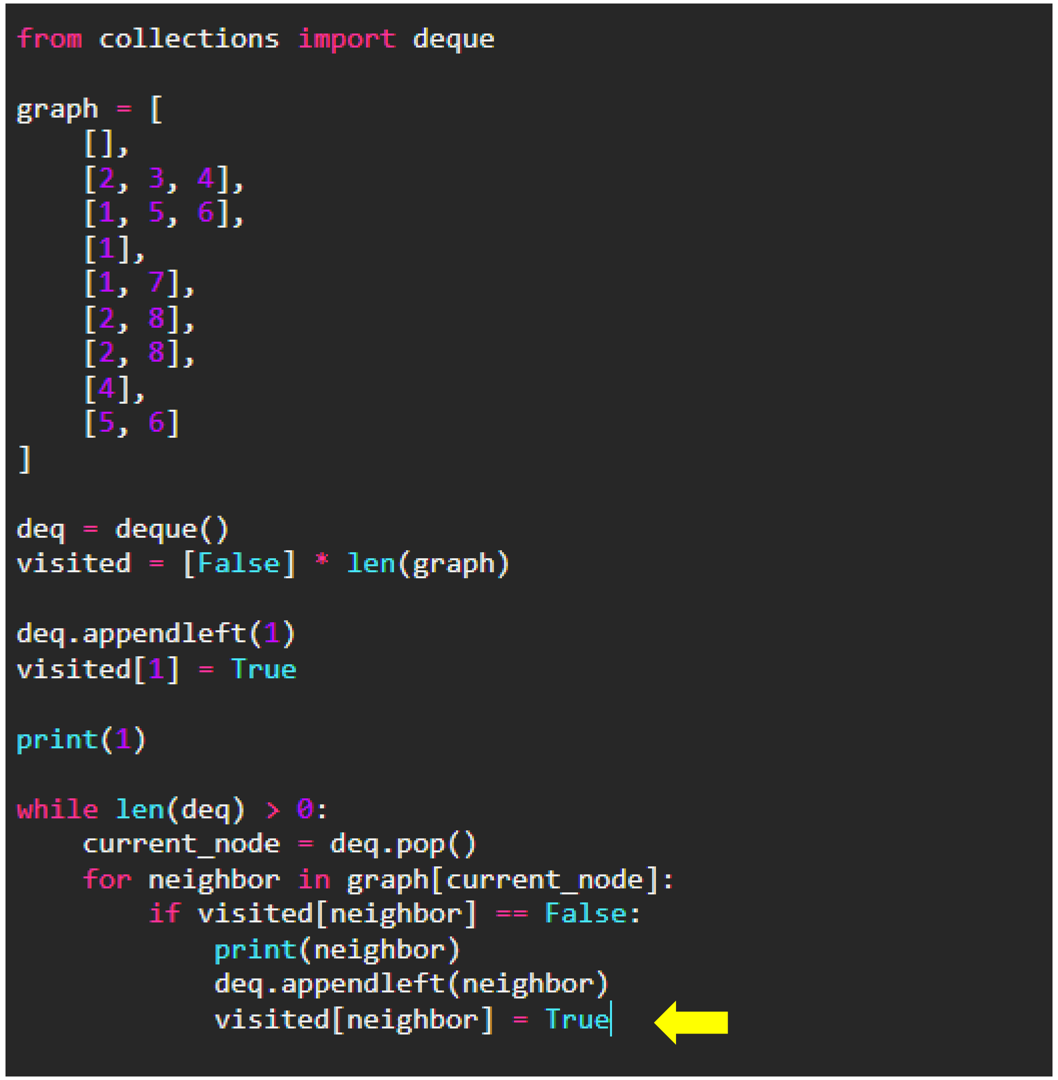




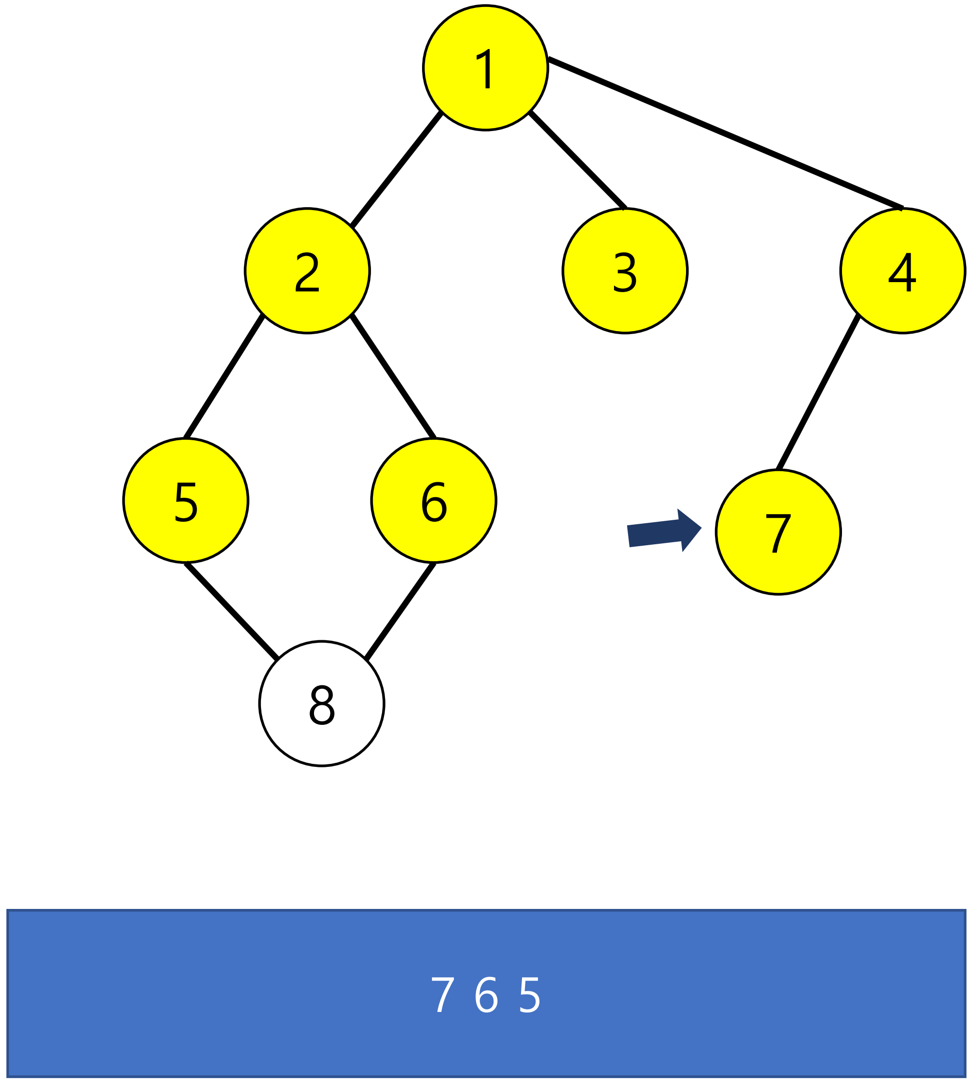
```
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```

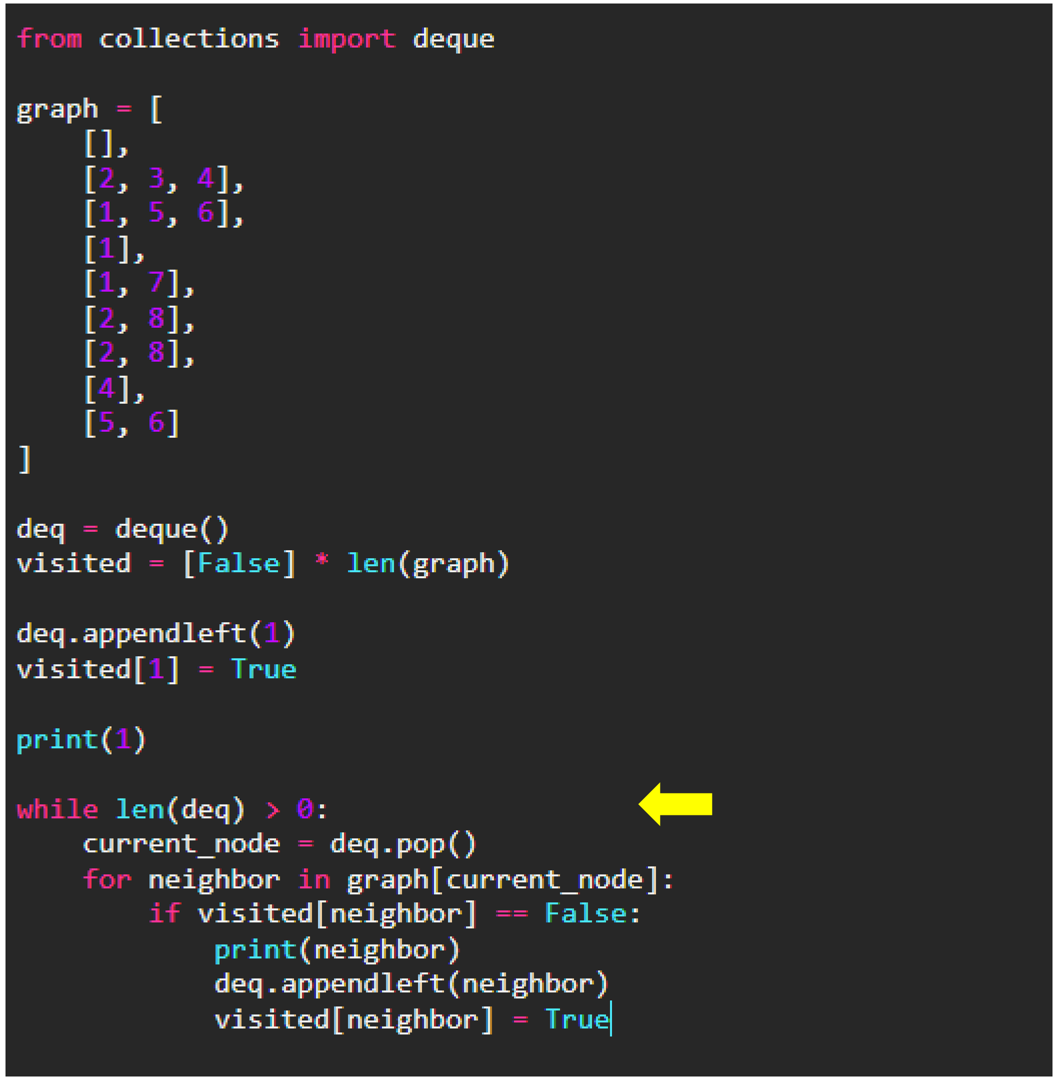




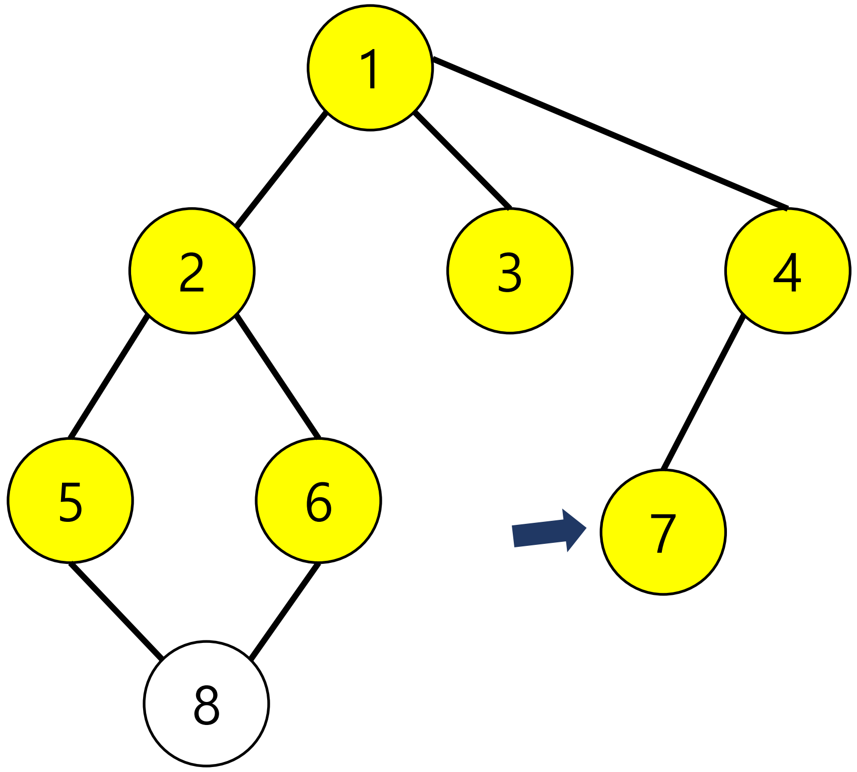


```
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```



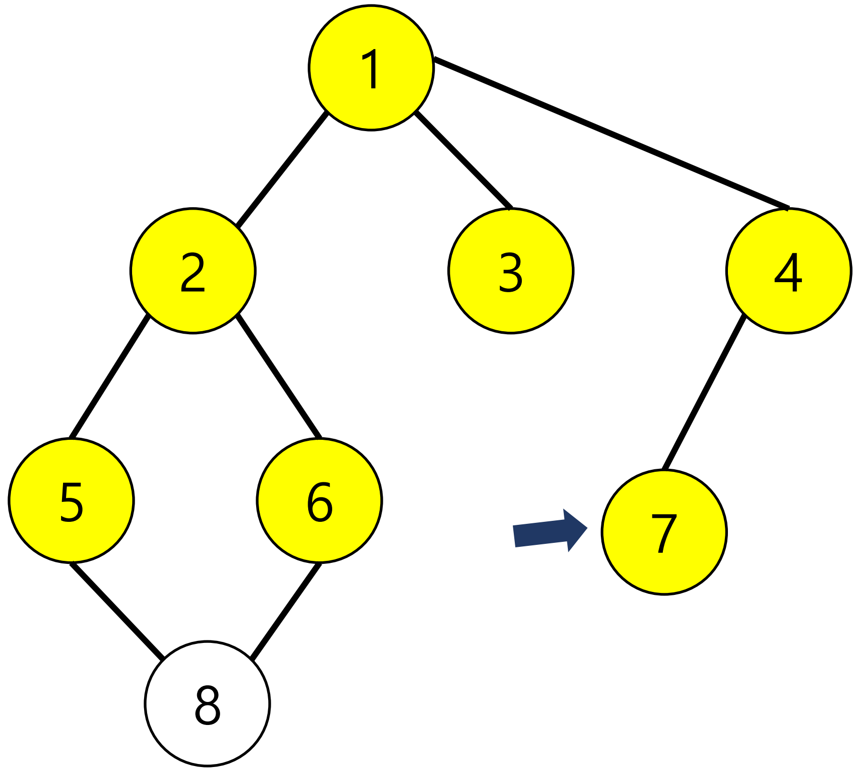


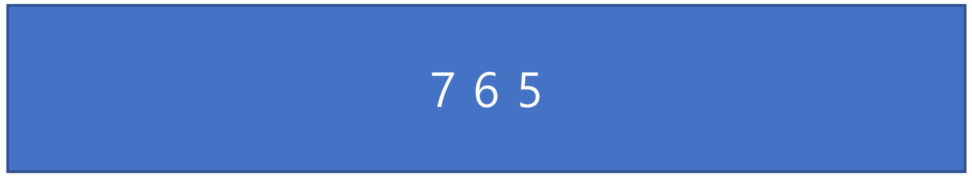
```
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```

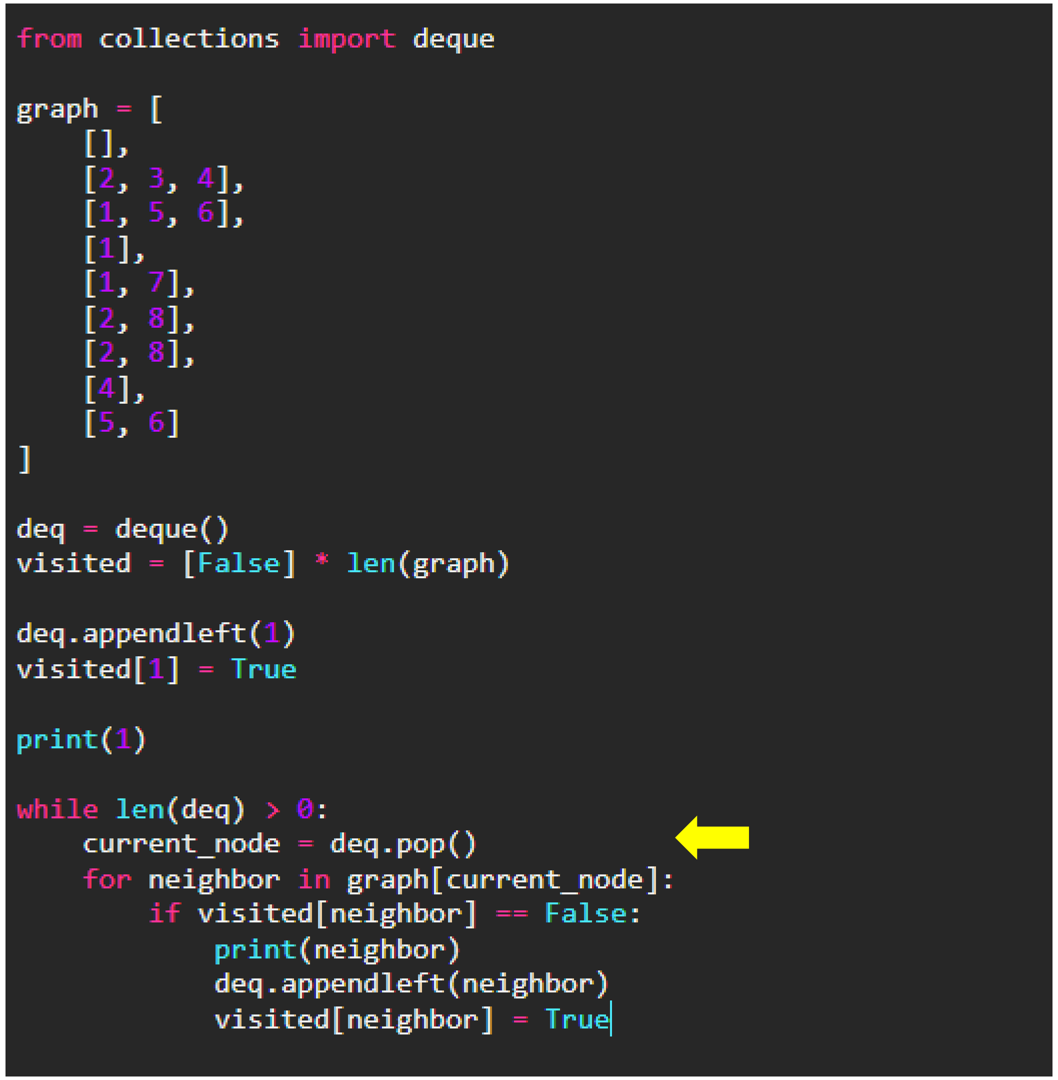


7 6 5

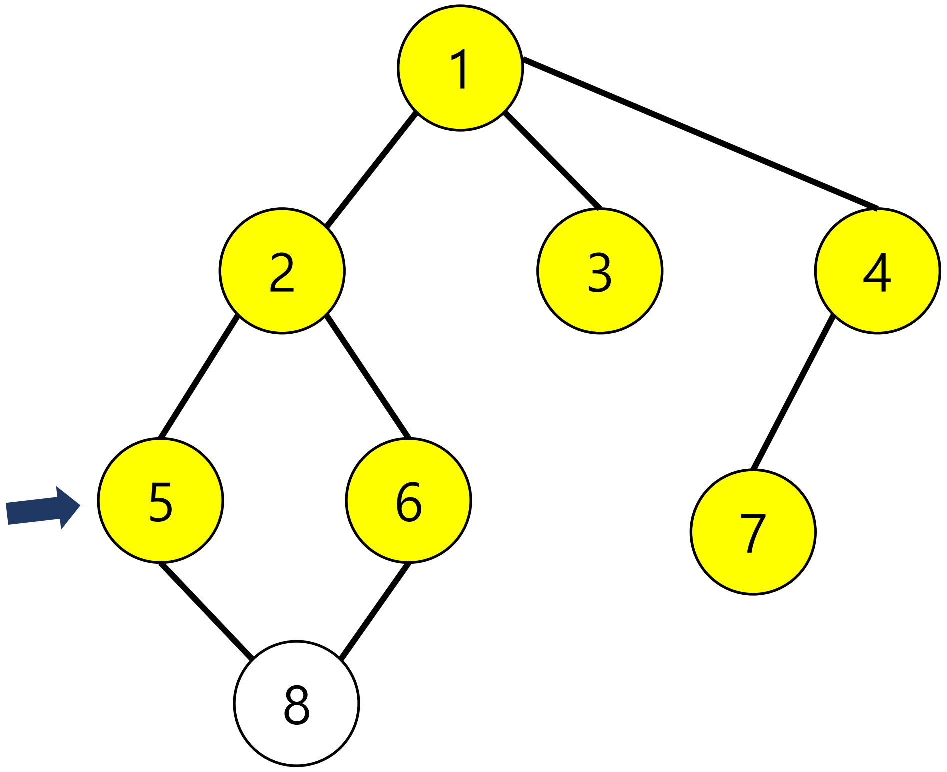
```
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```

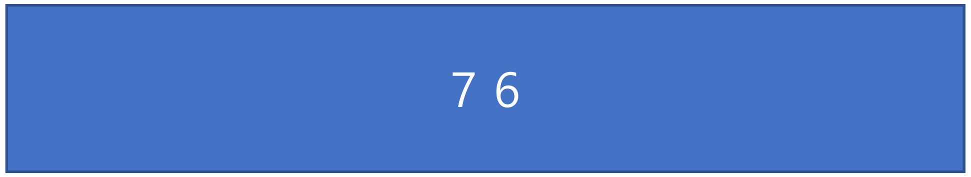


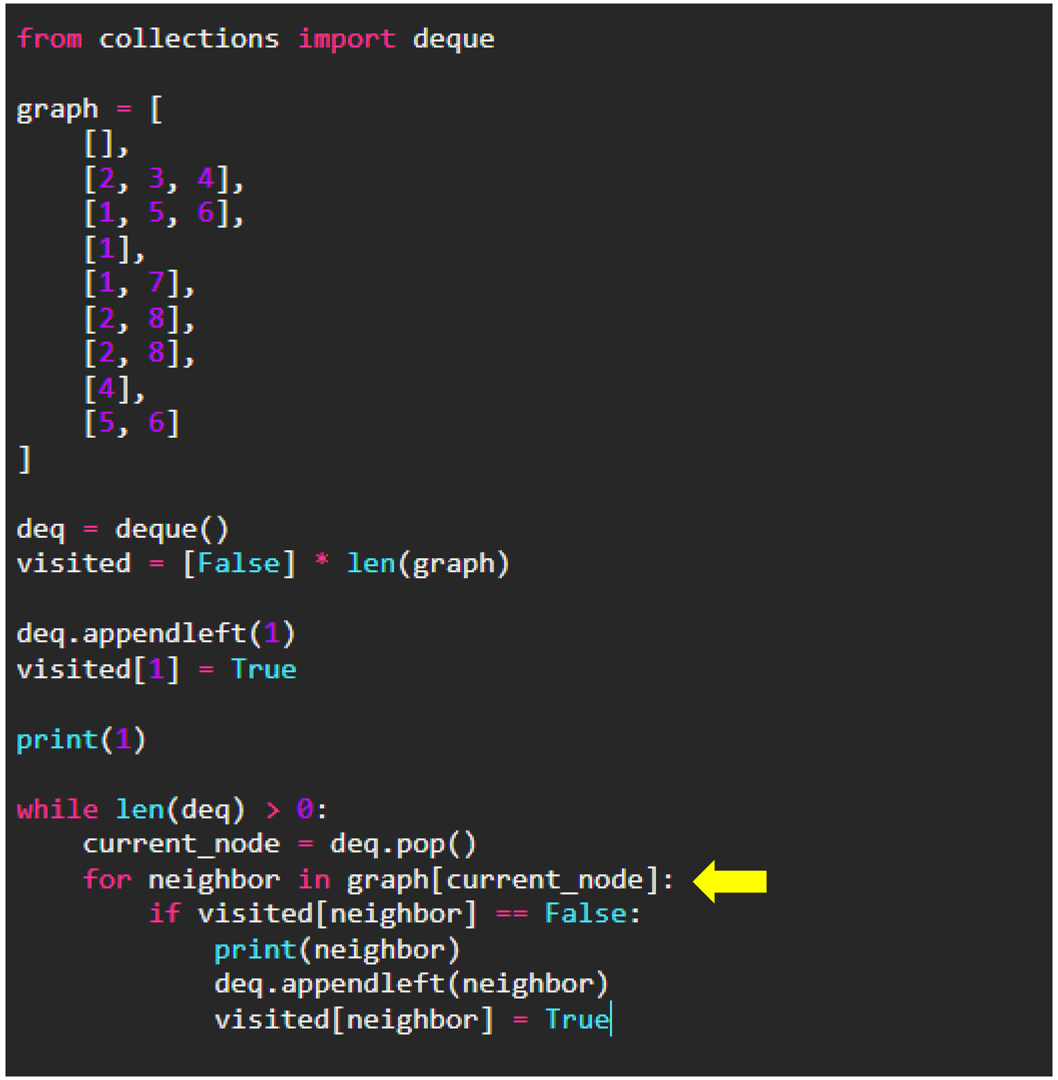




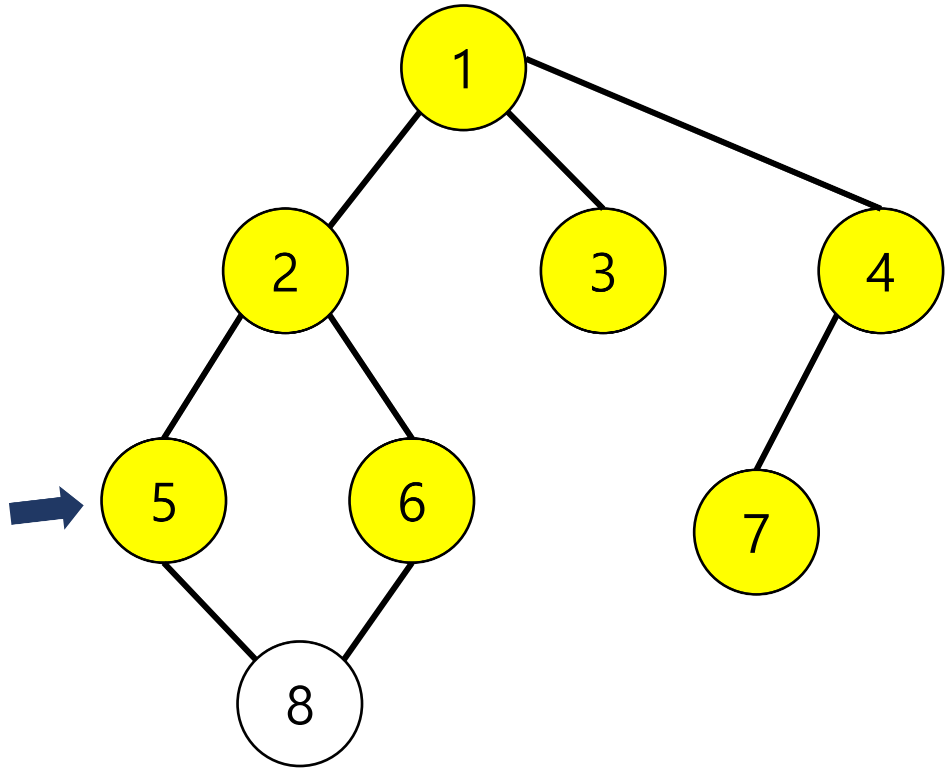
```
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```

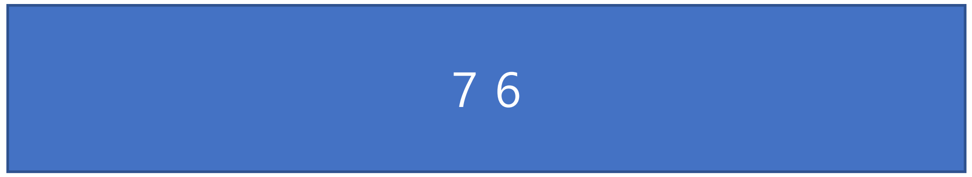


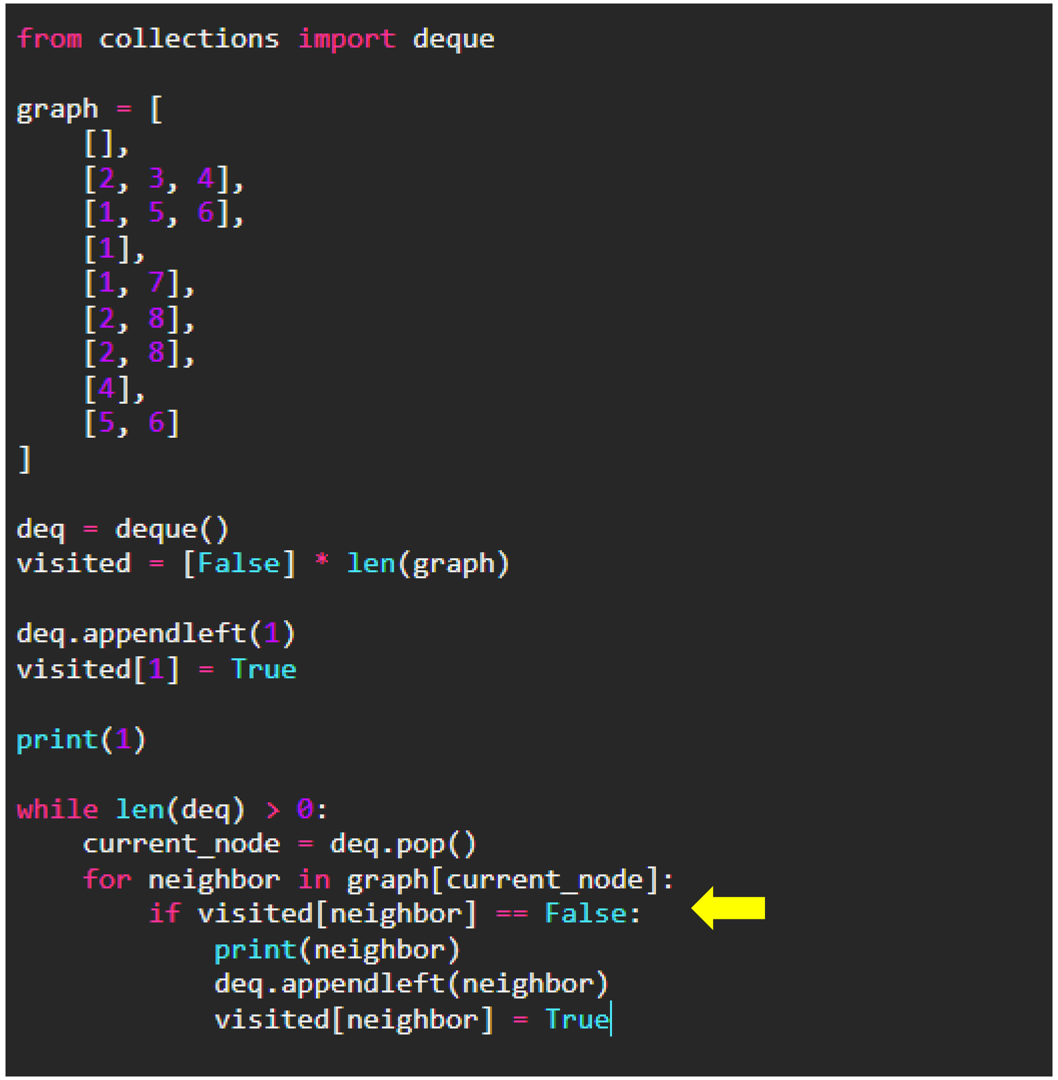




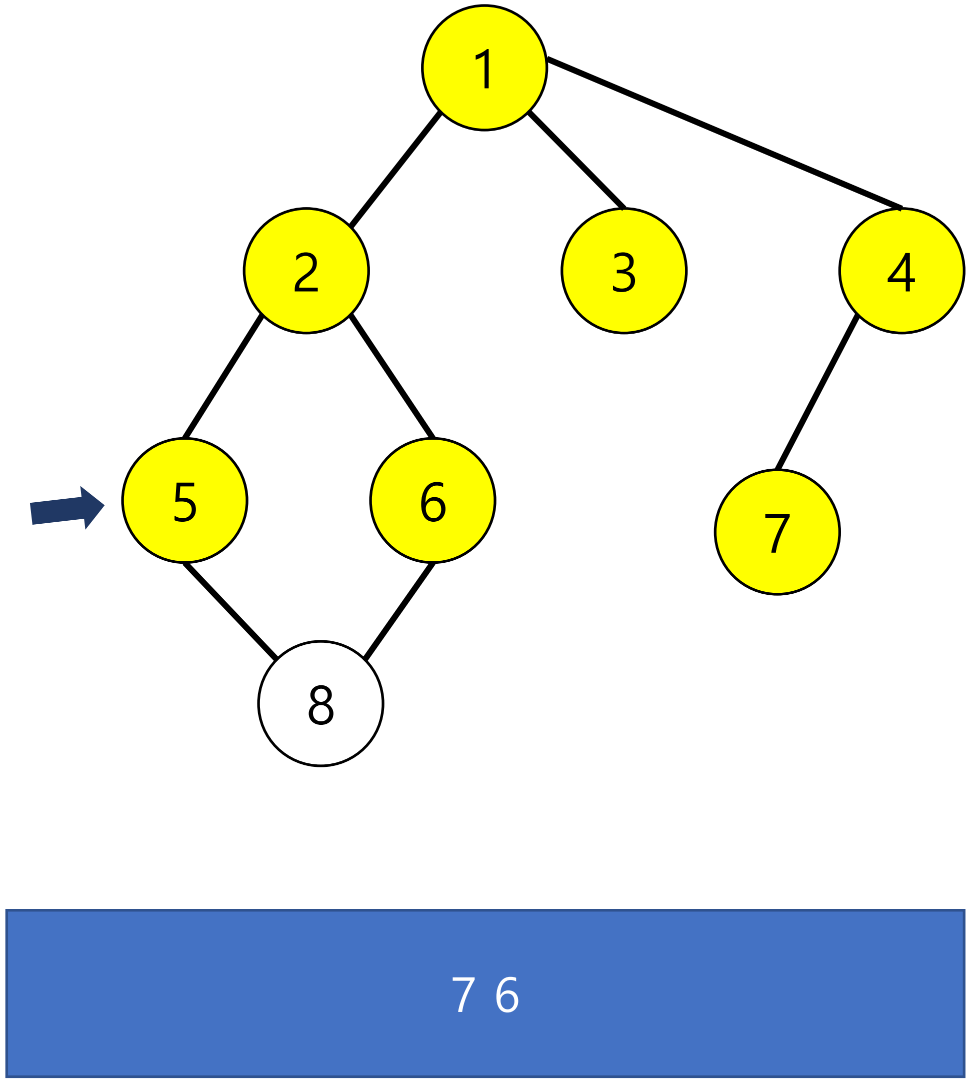
```
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```

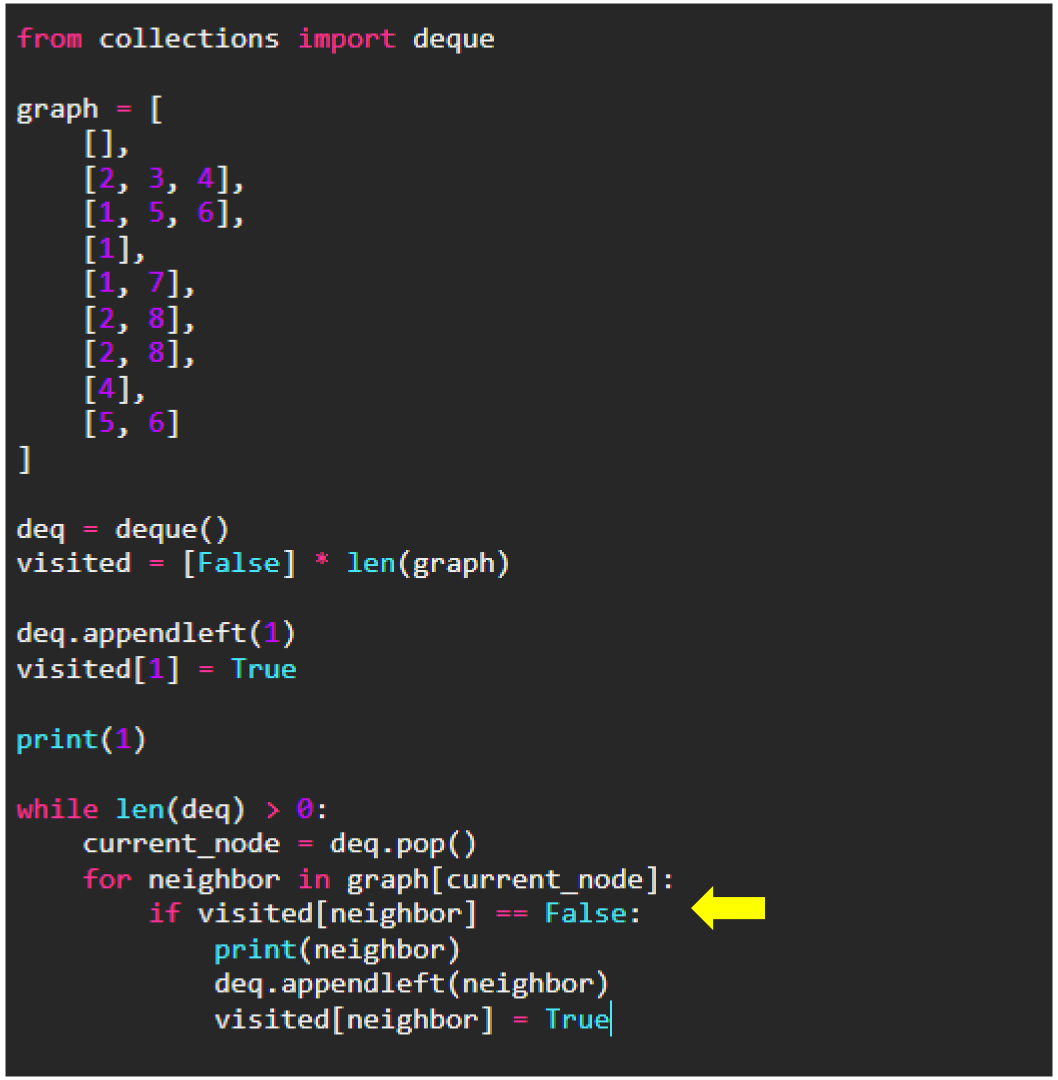




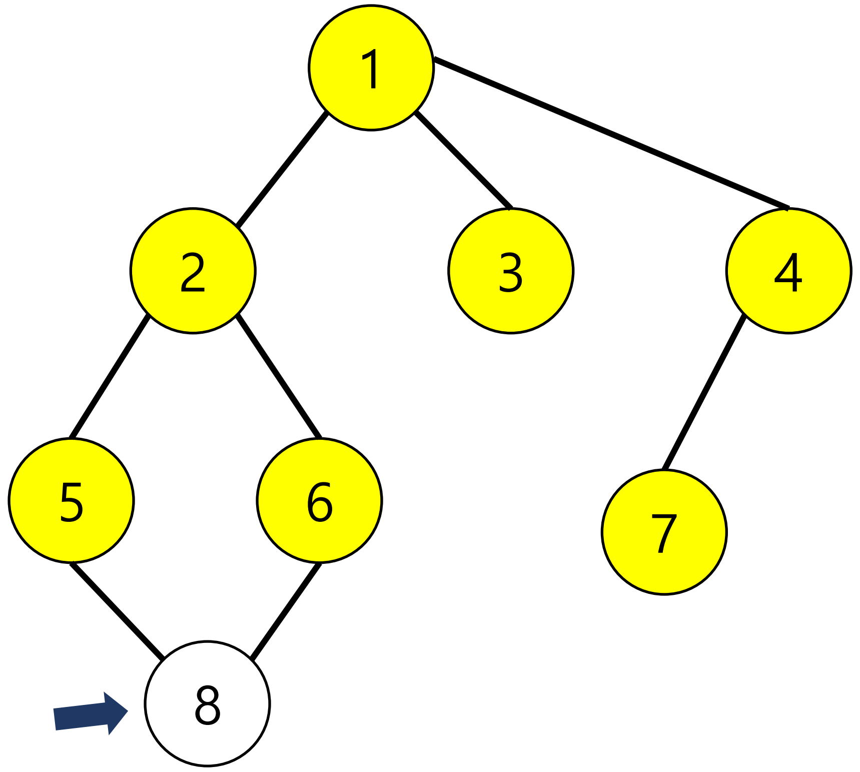


```python
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```

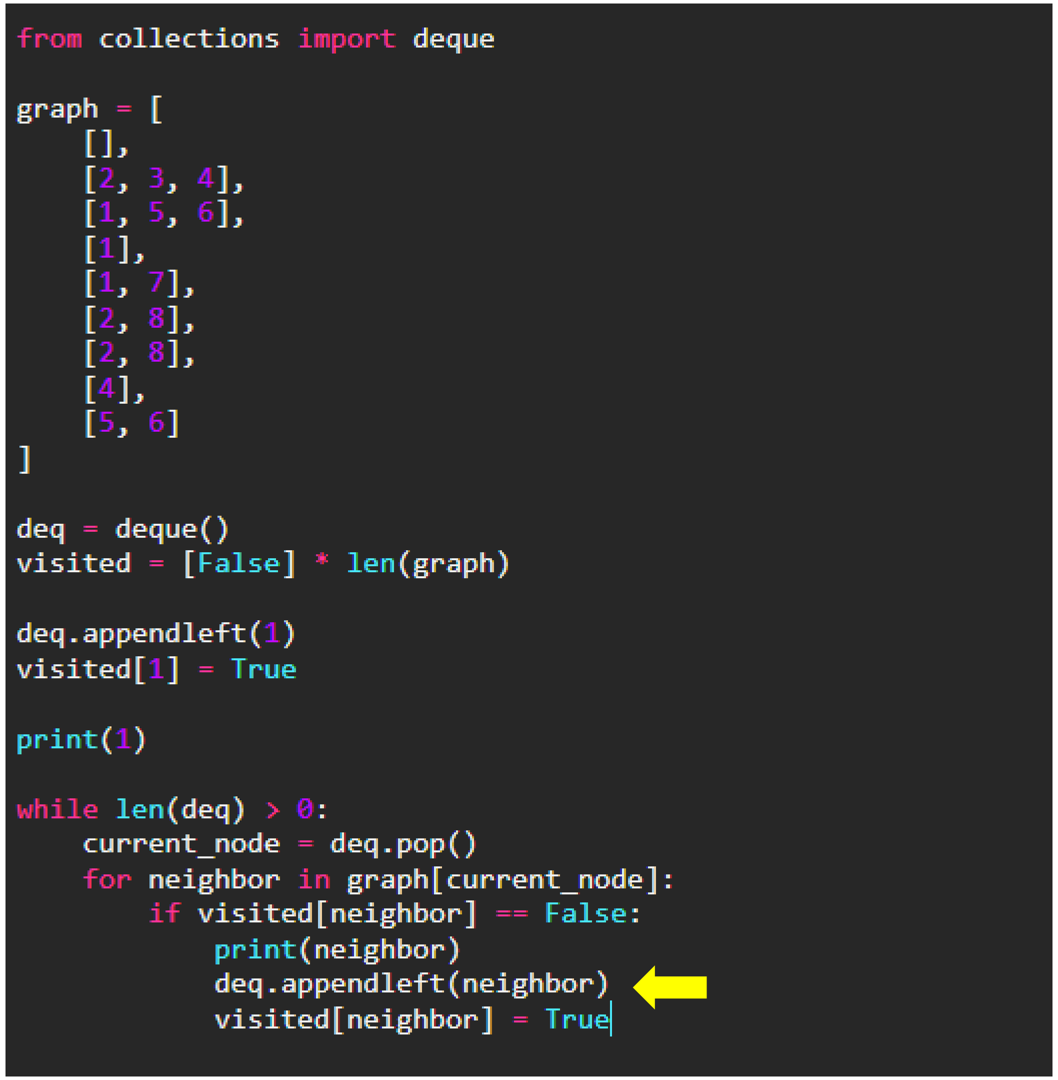




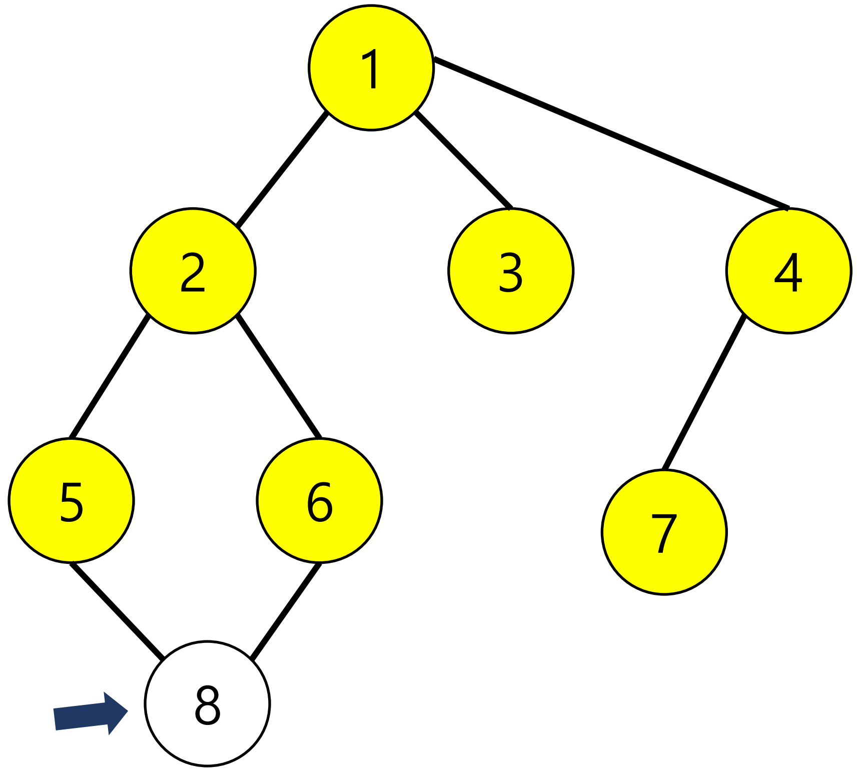
```
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```

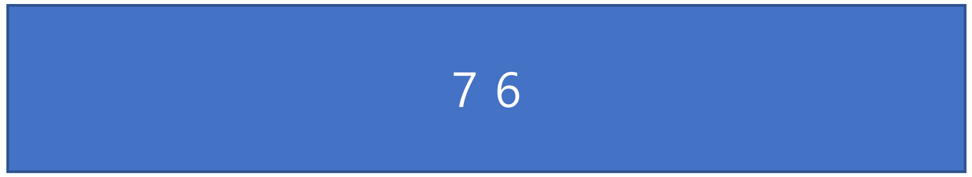


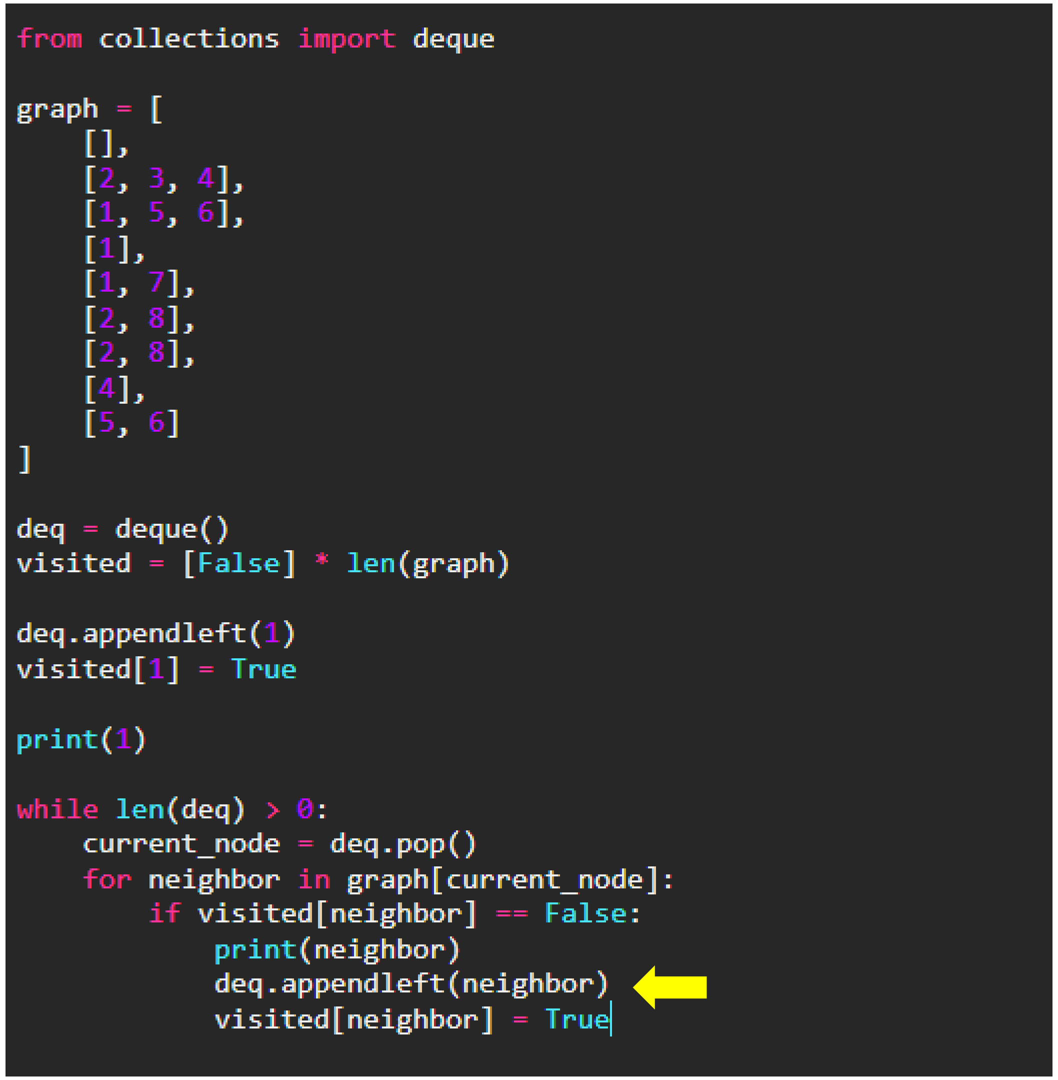
7 6



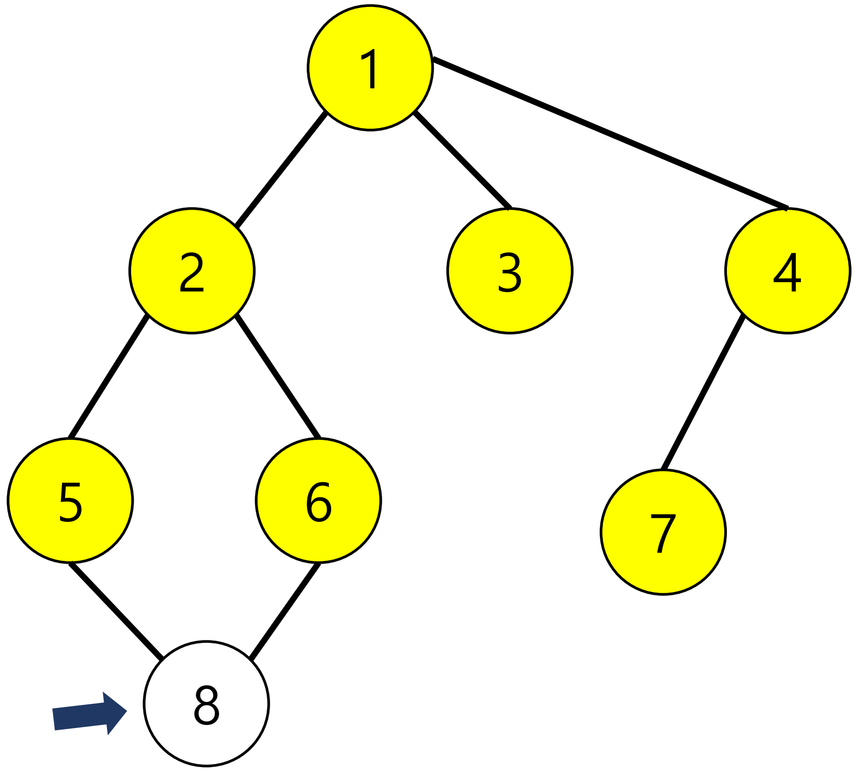
```
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```

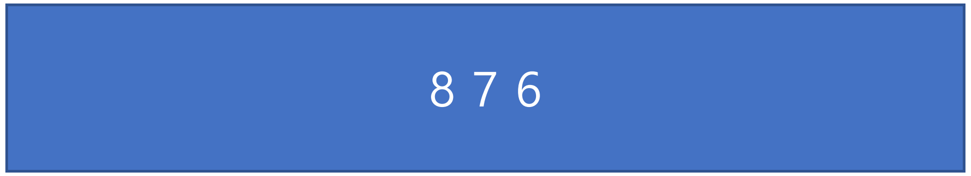


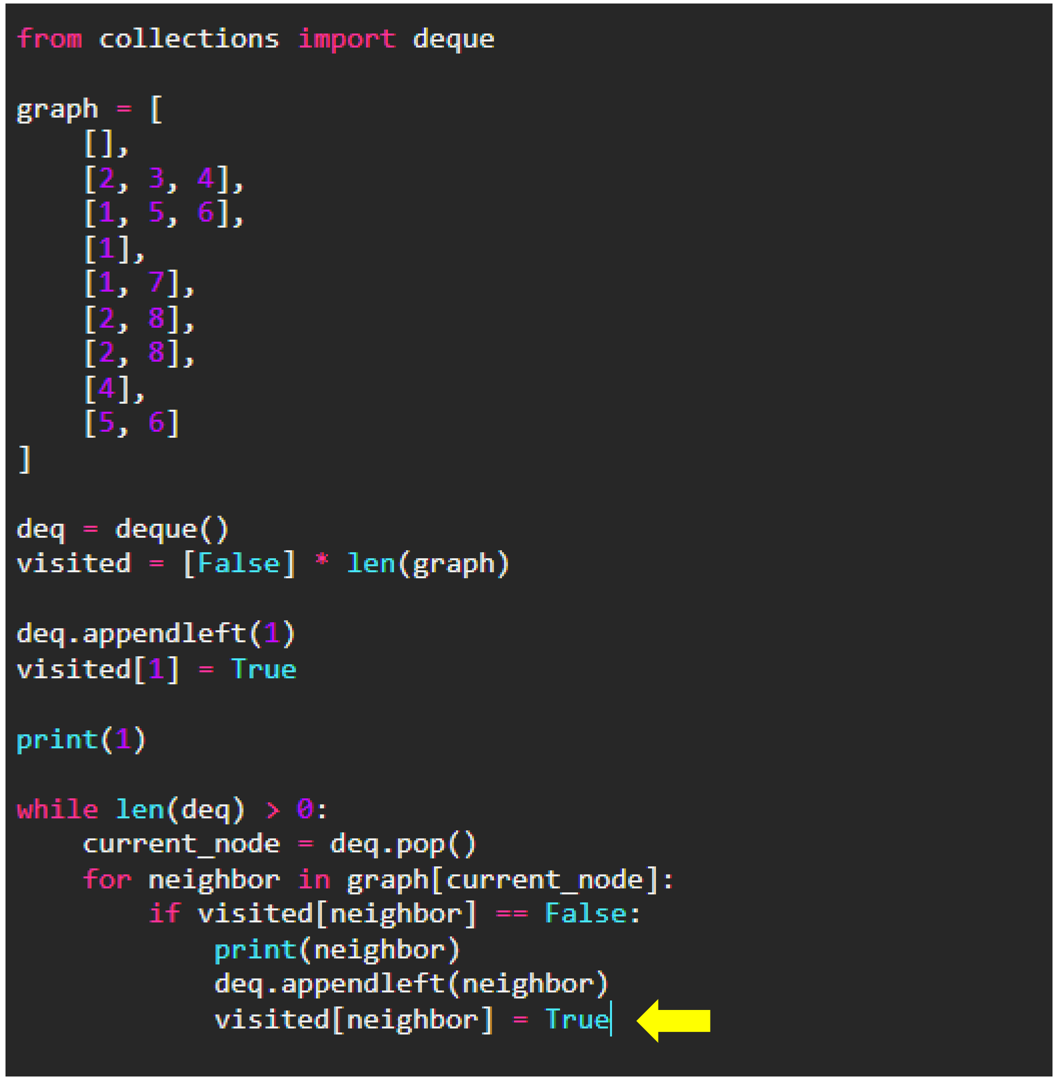


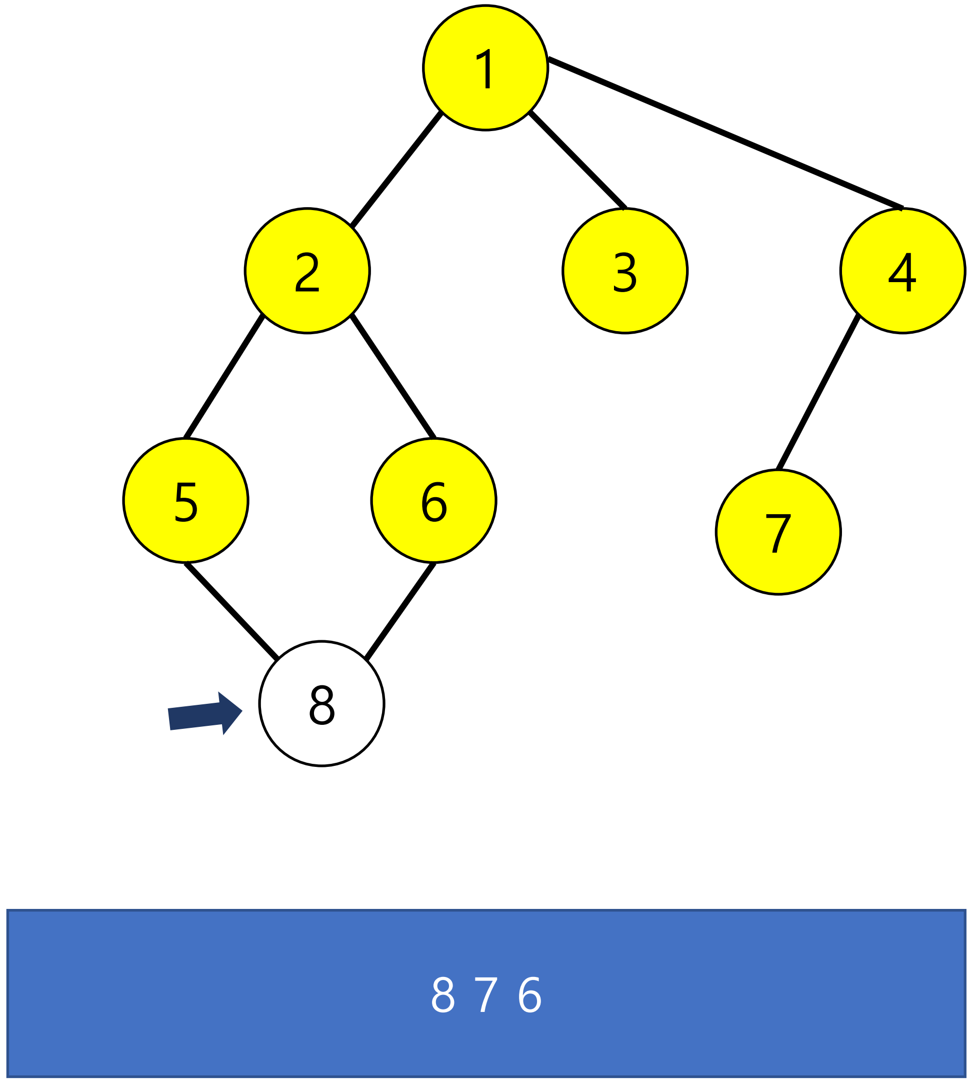


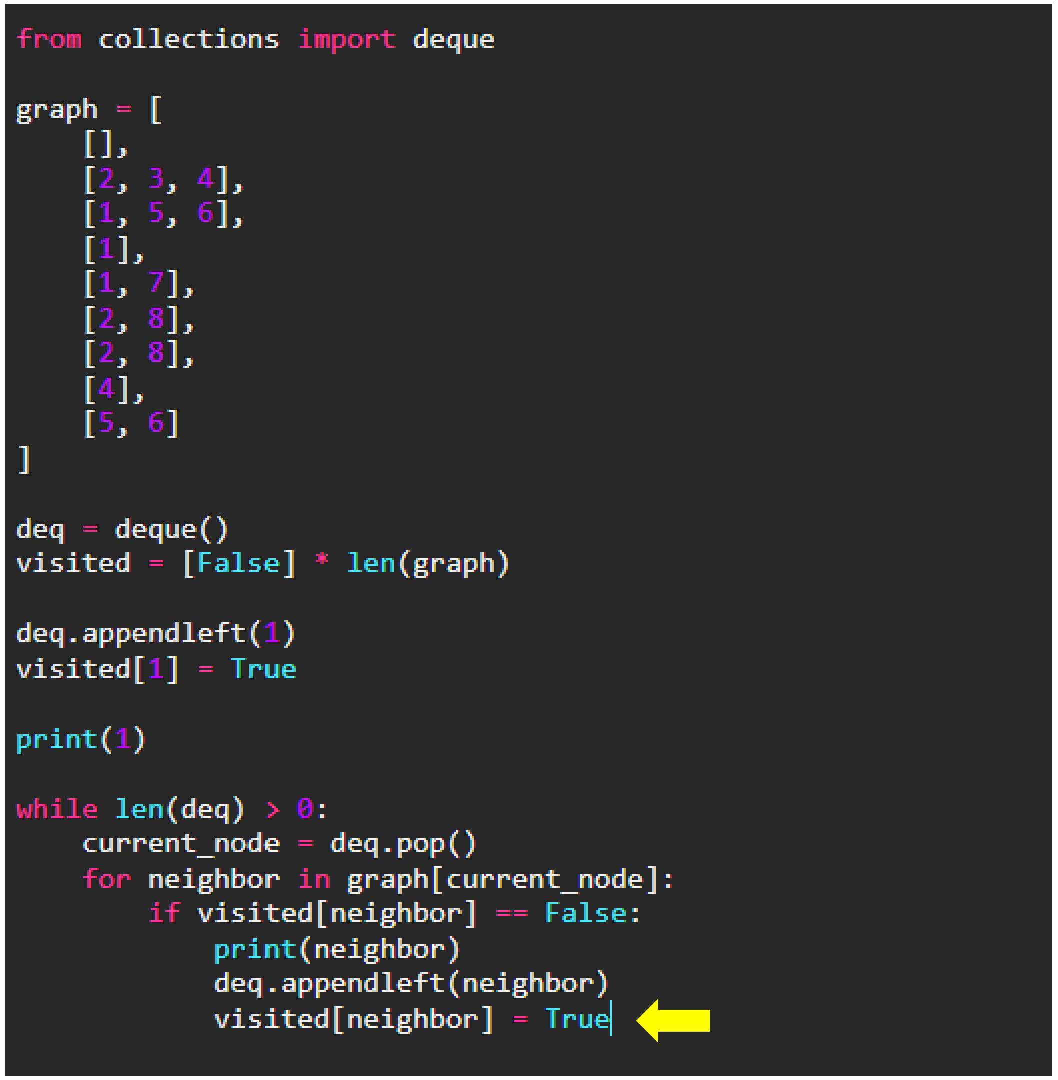
```
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```



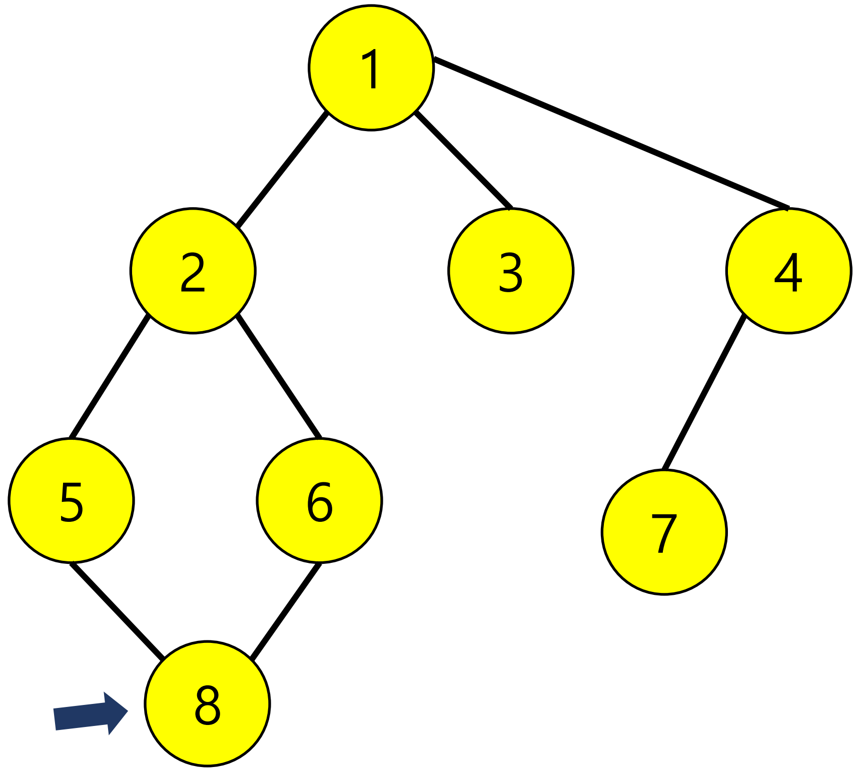


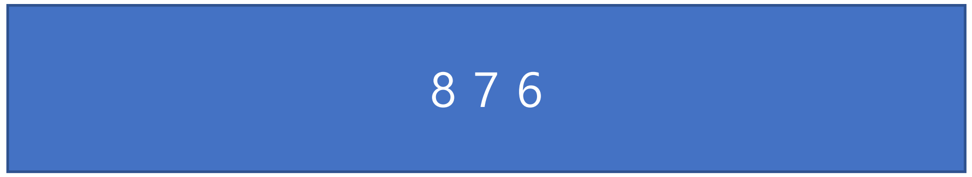

```python
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```

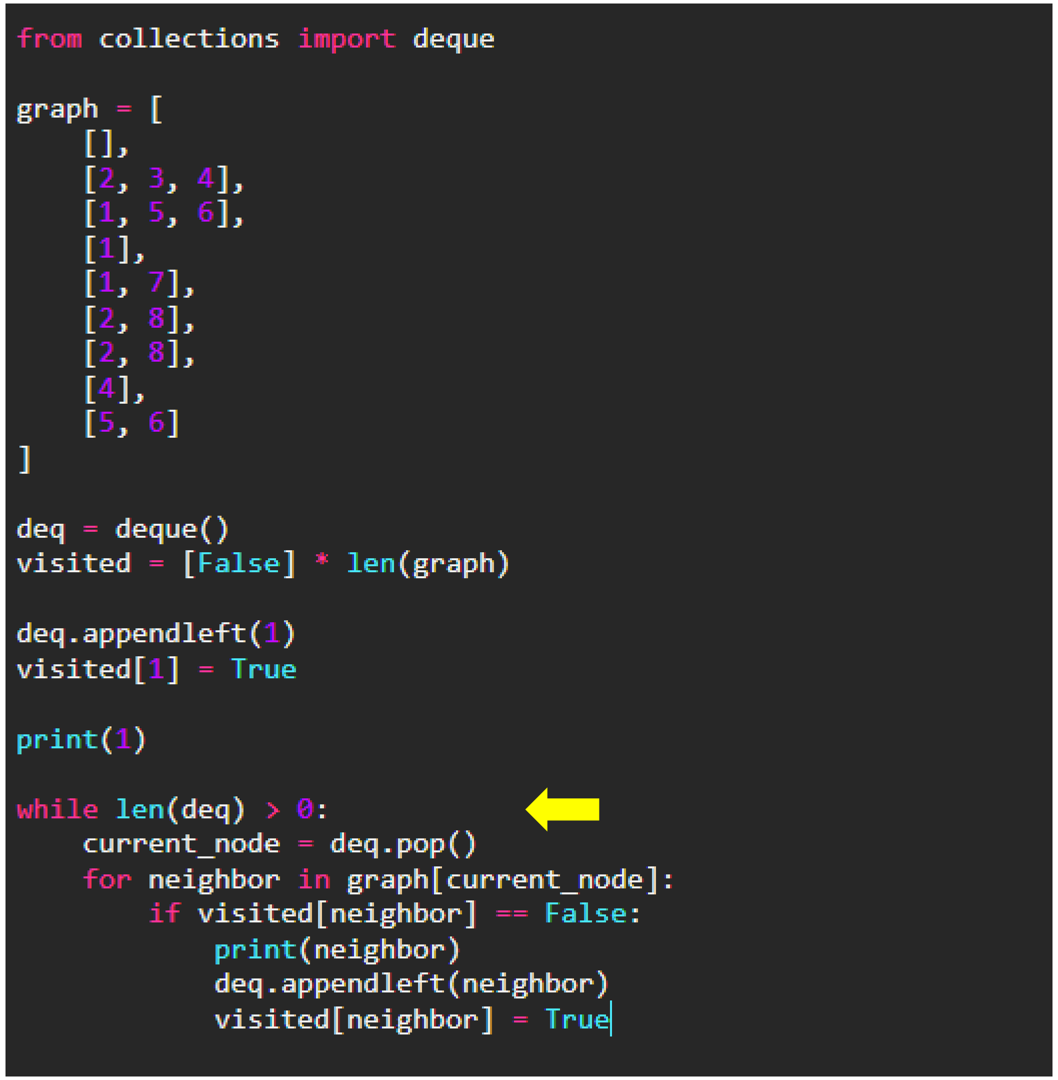




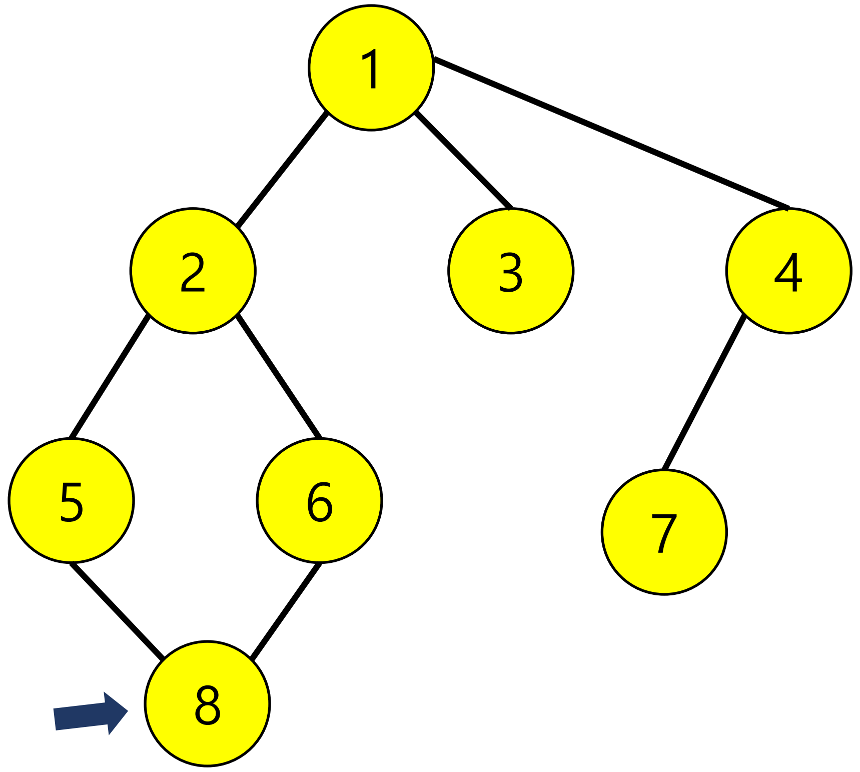
```
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```

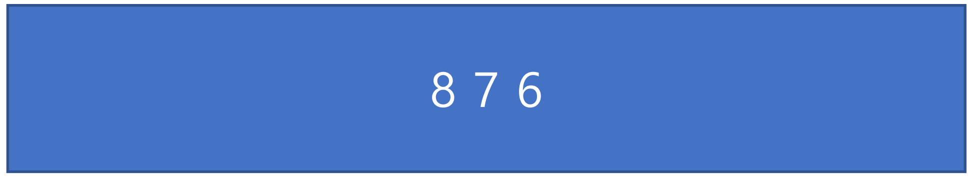


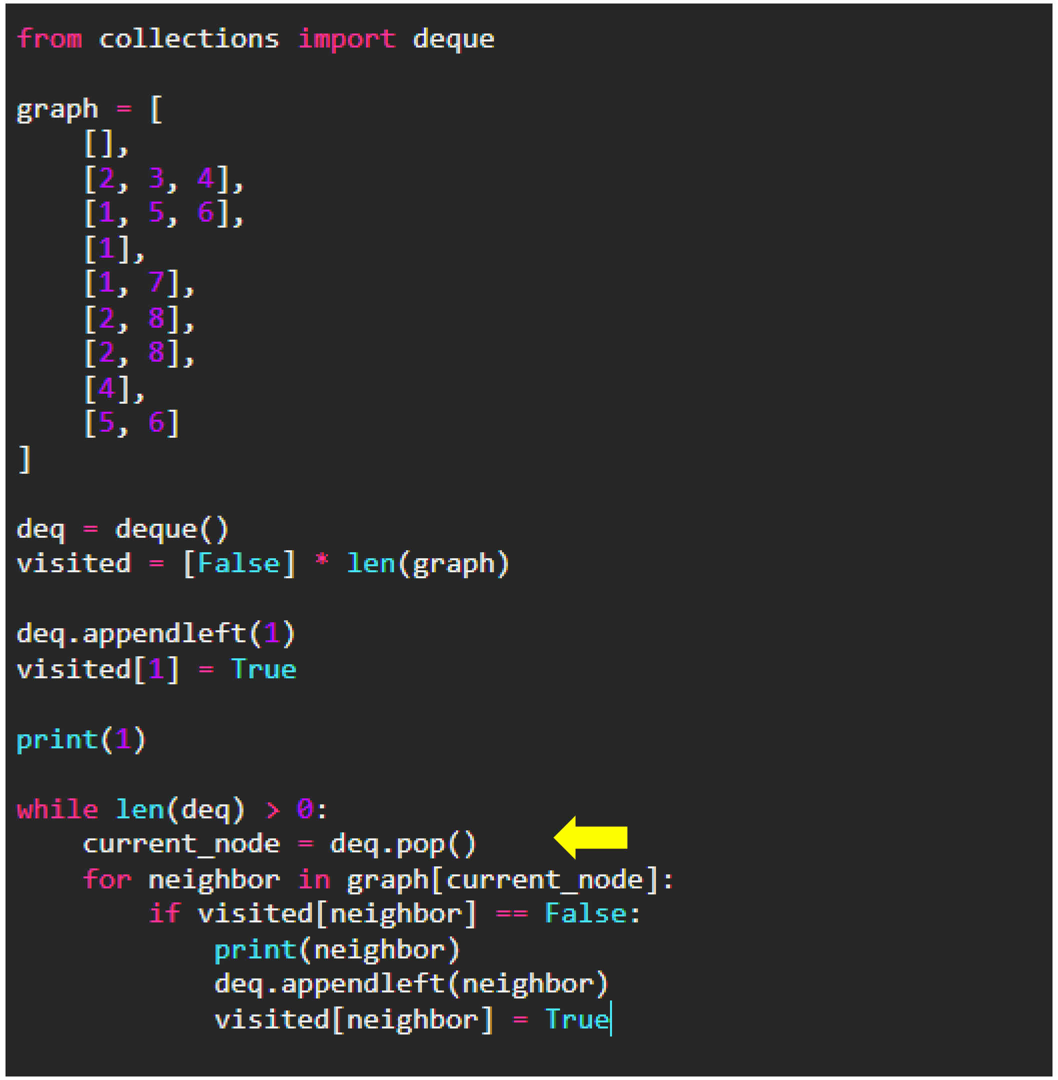




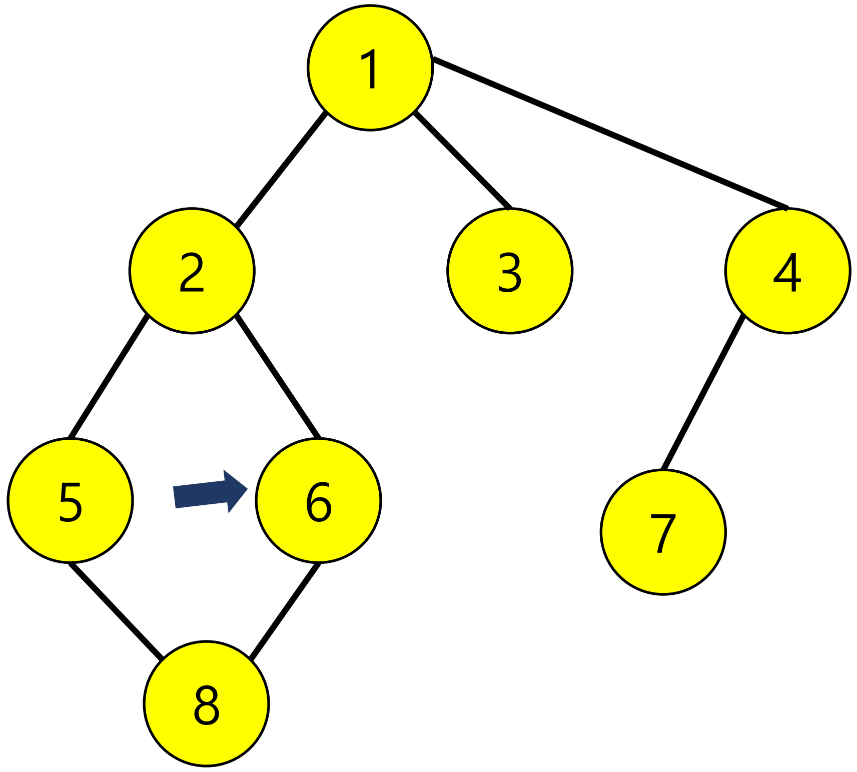
```
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```



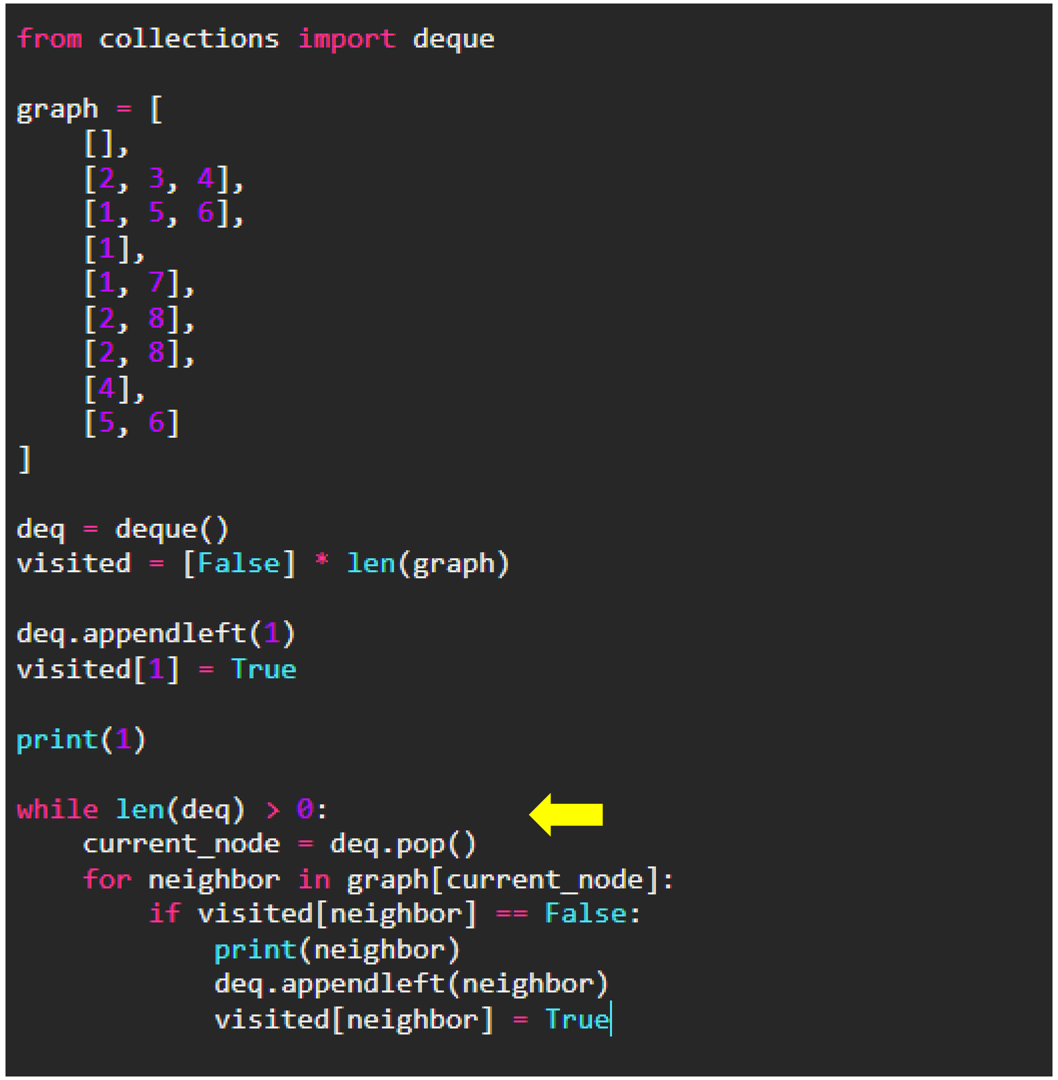




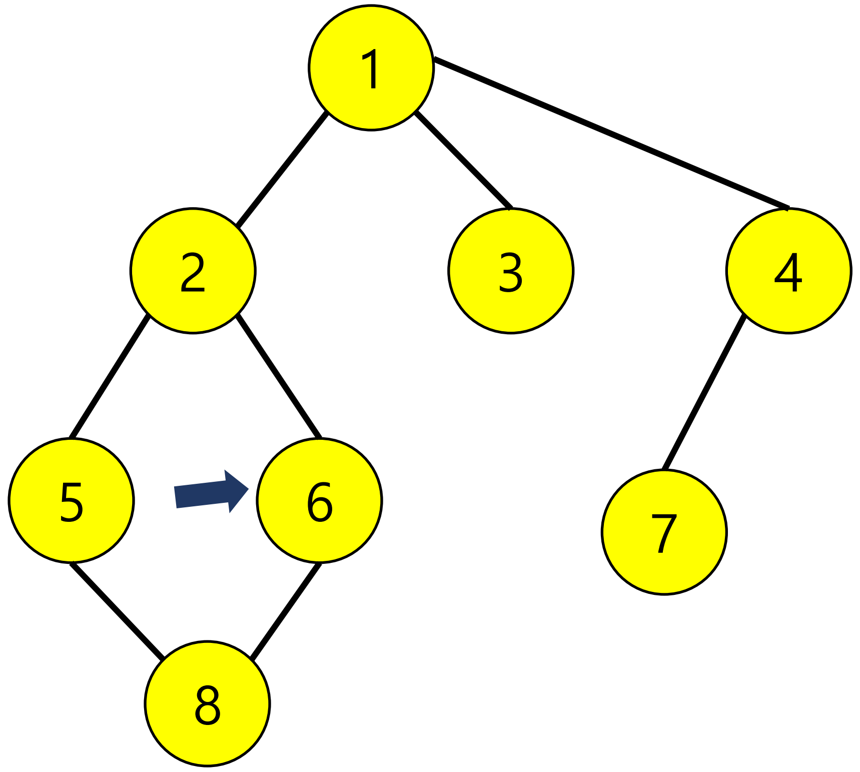
```
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```

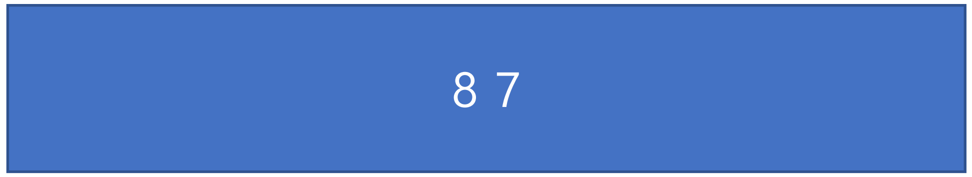


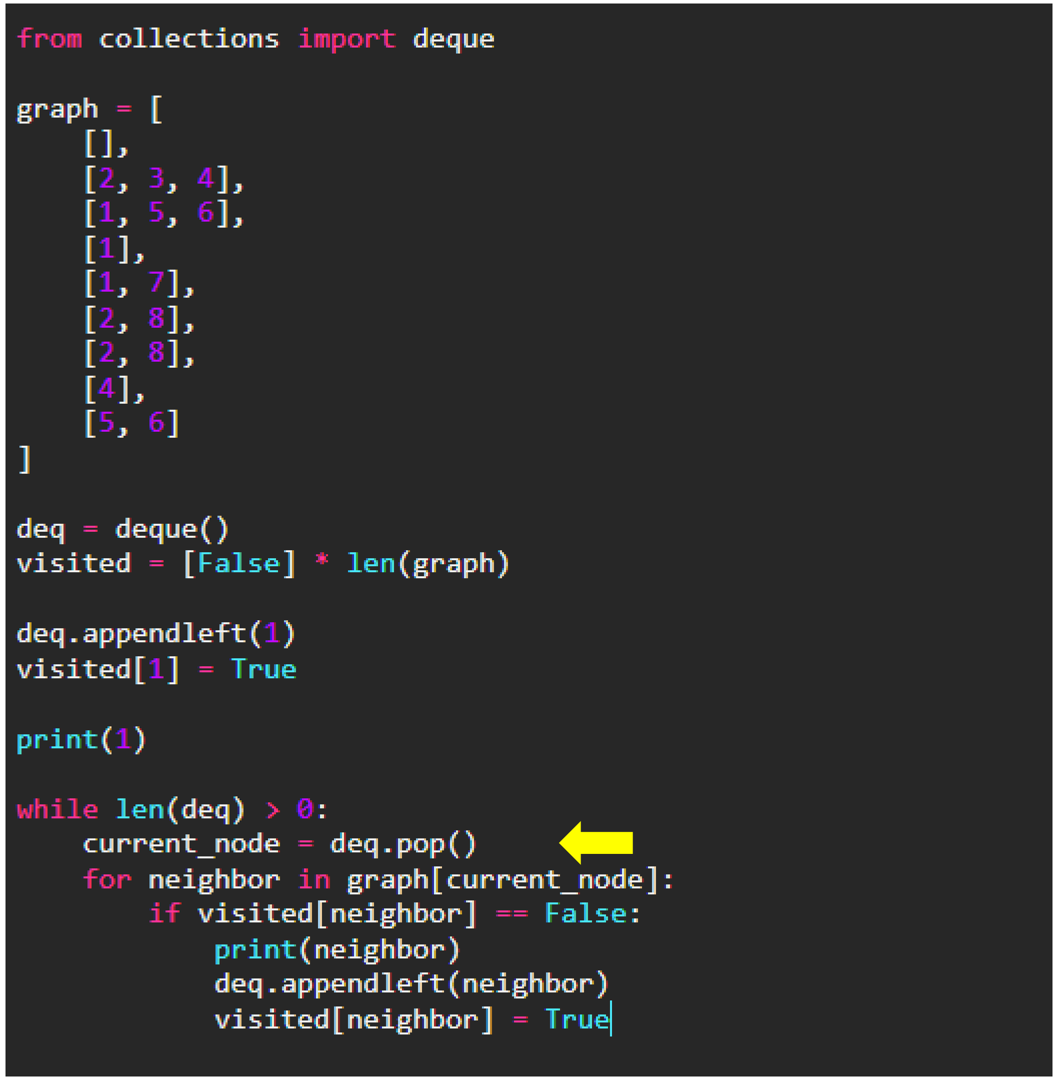
8 7



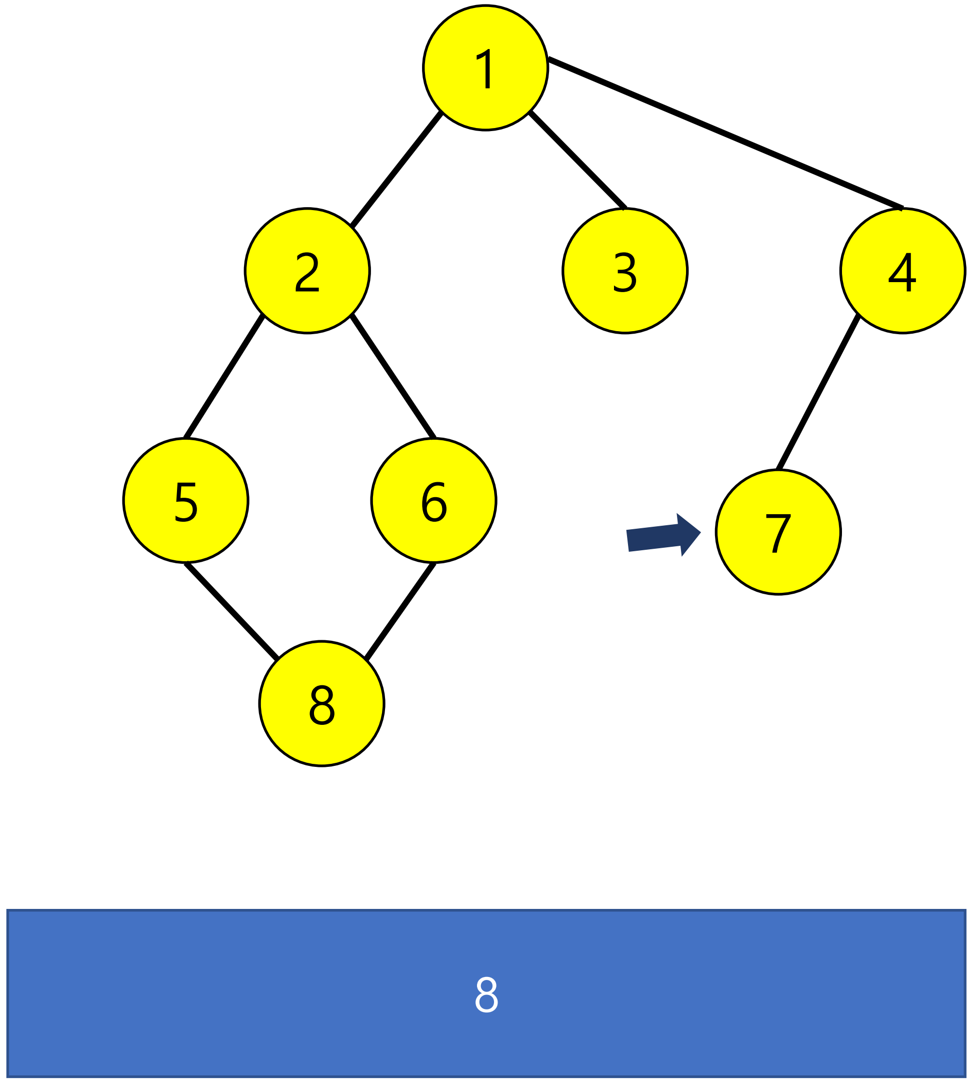
```
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```

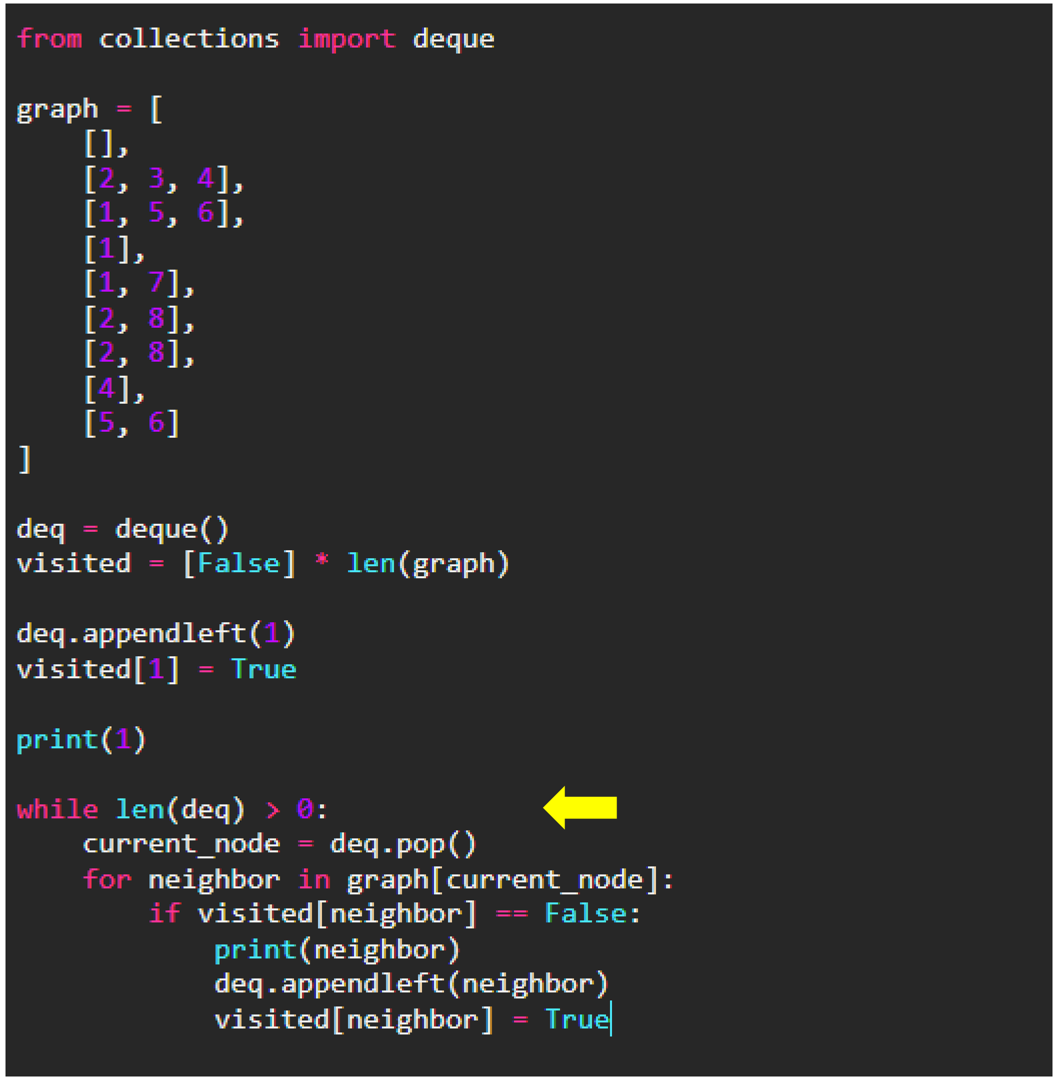




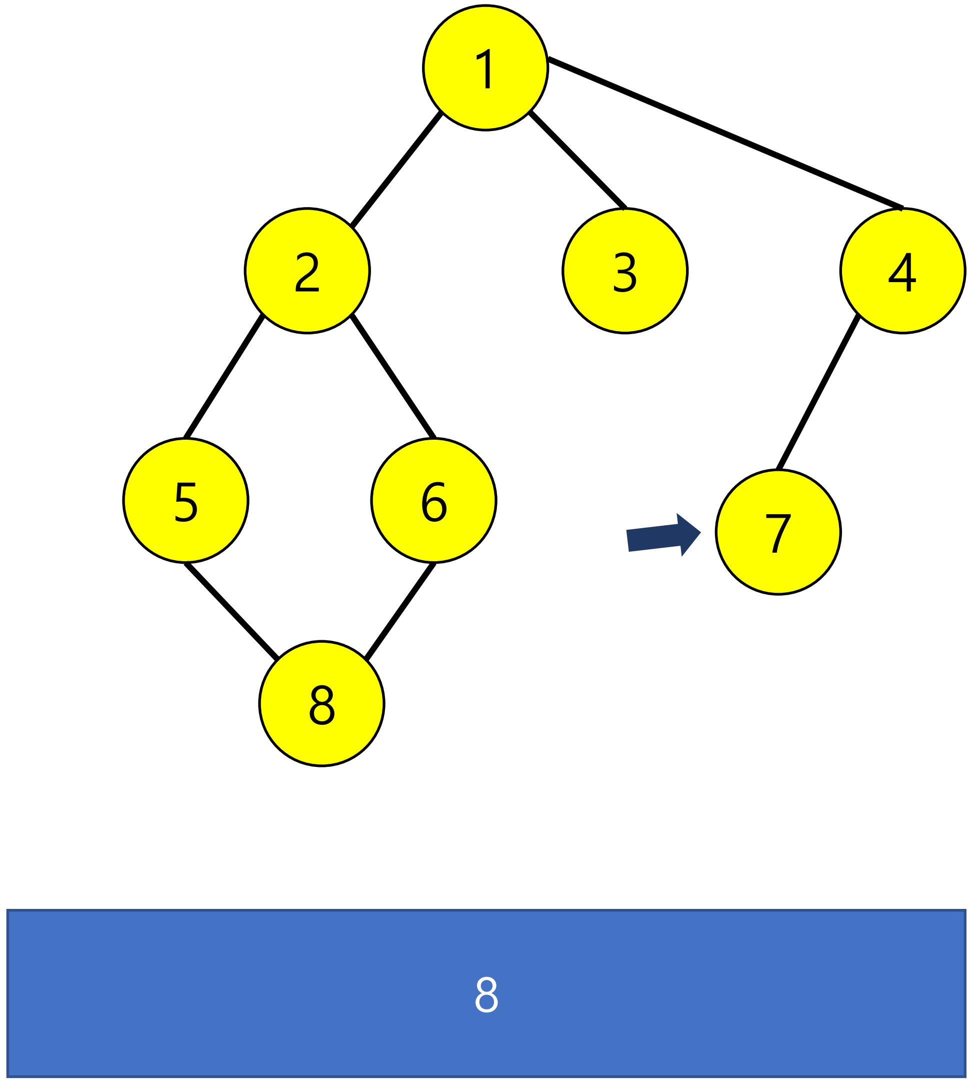


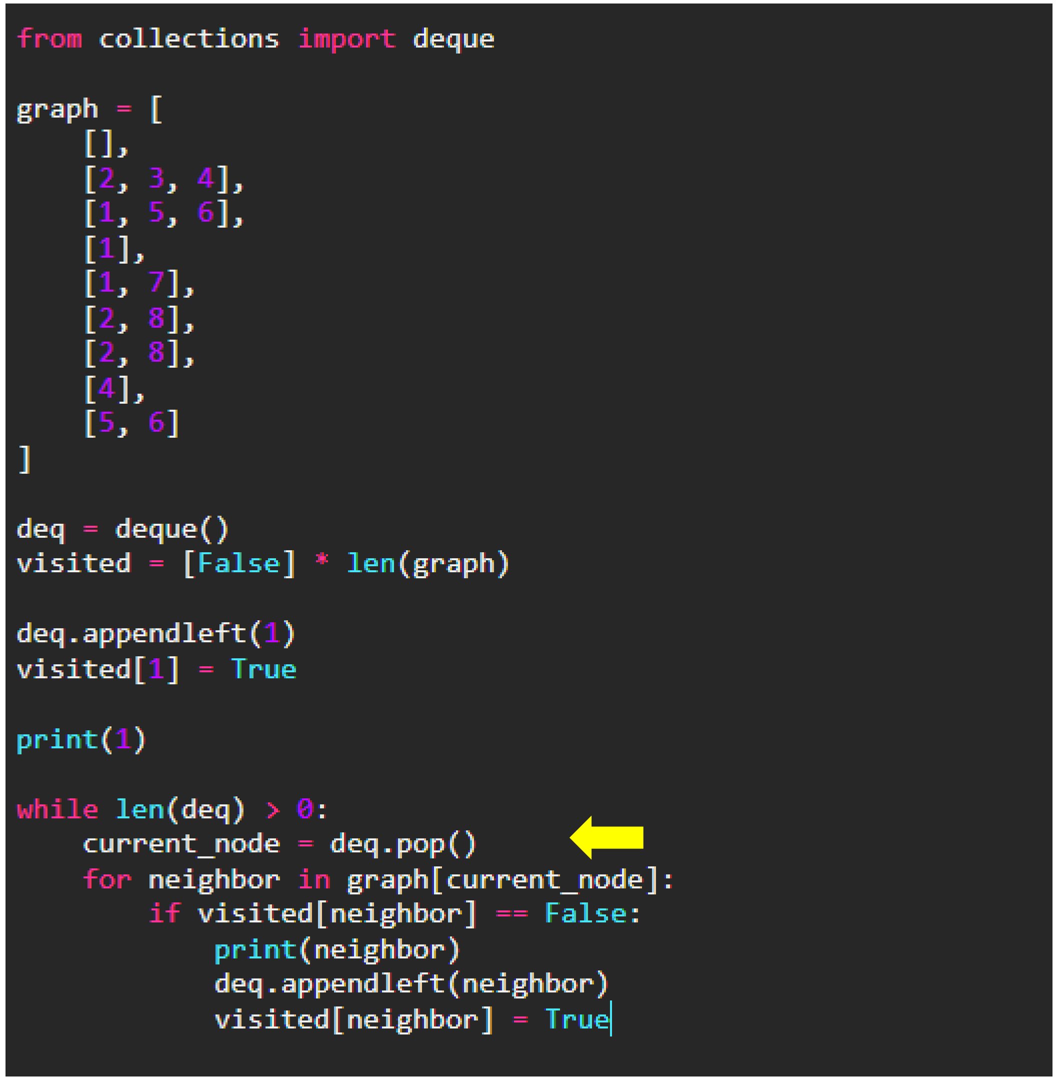
```python
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```



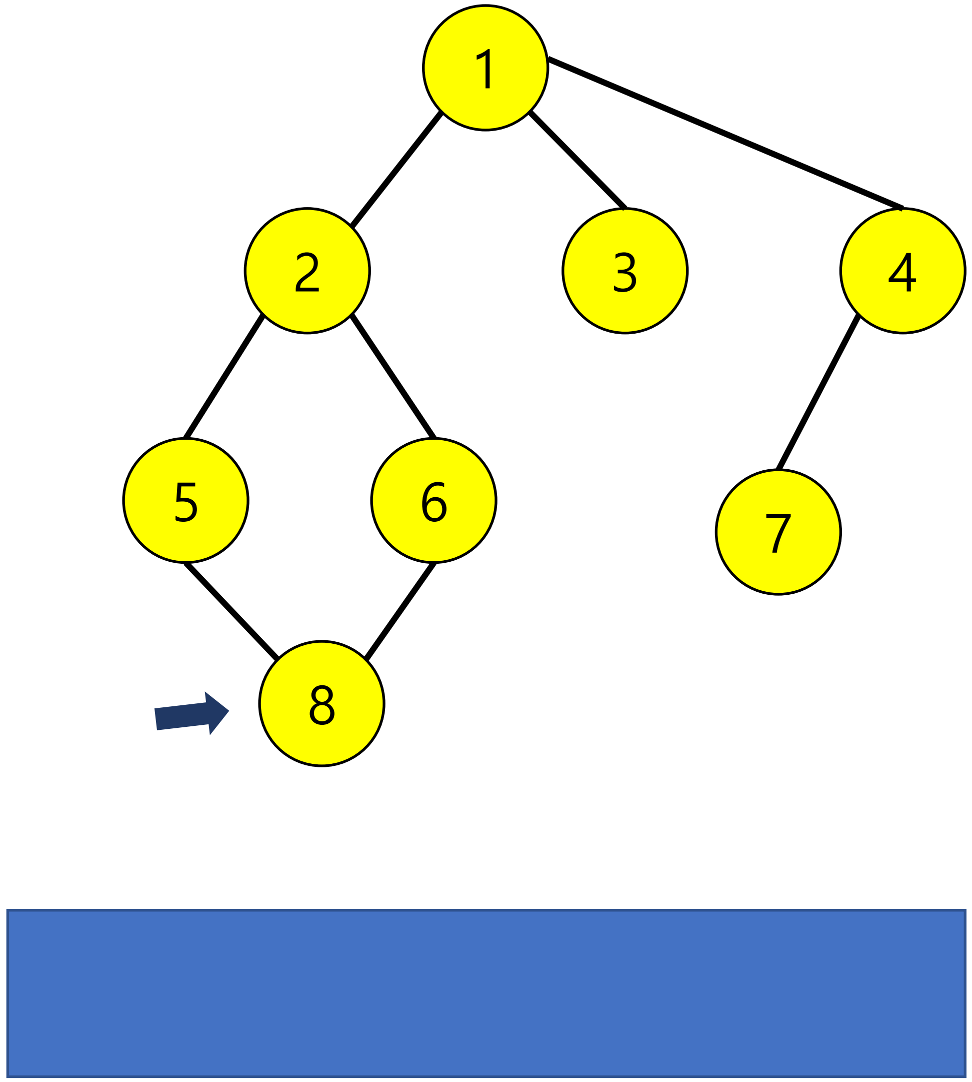


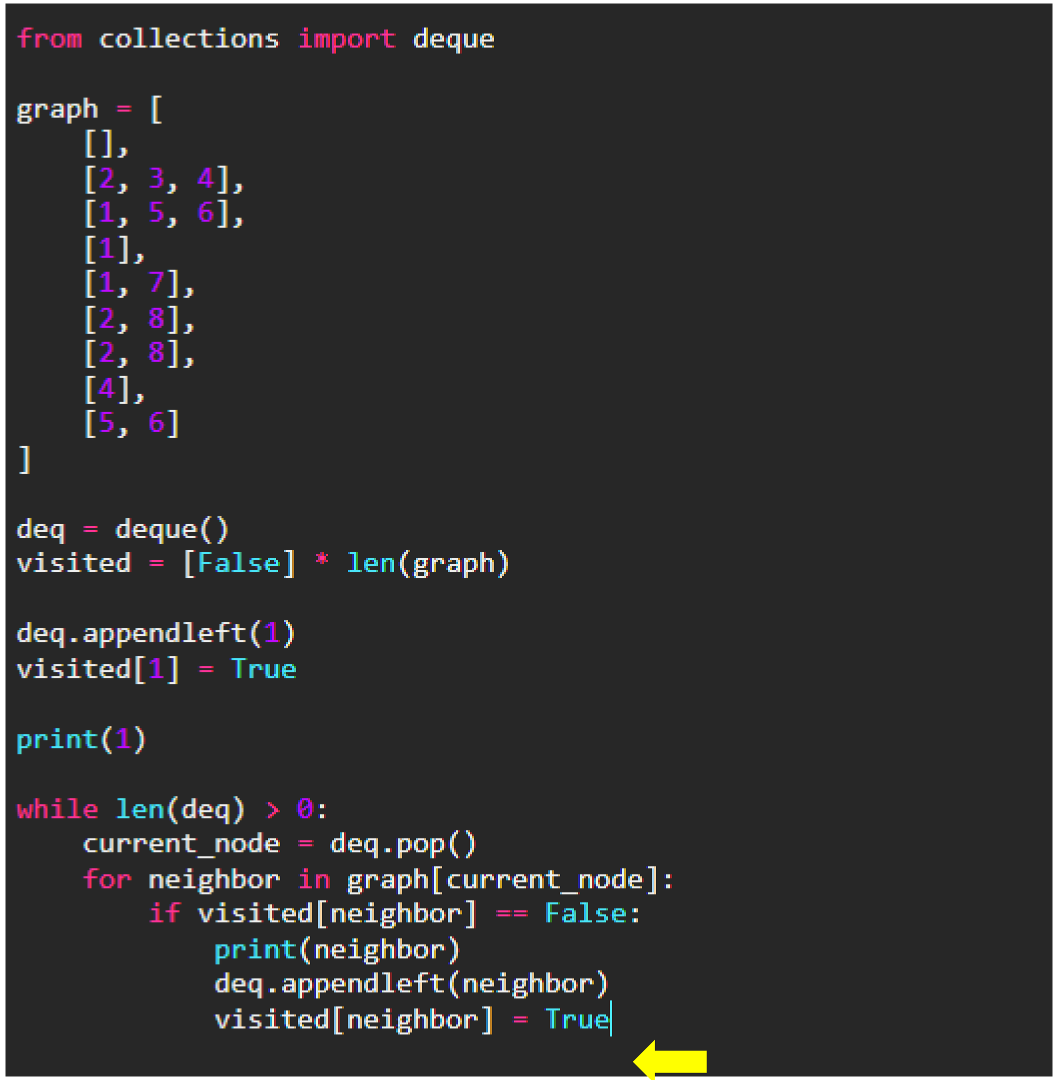
```
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```





```
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```





```
from collections import deque

graph = [
    [],
    [2, 3, 4],
    [1, 5, 6],
    [1],
    [1, 7],
    [2, 8],
    [2, 8],
    [4],
    [5, 6]
]

deq = deque()
visited = [False] * len(graph)

deq.appendleft(1)
visited[1] = True

print(1)

while len(deq) > 0:
    current_node = deq.pop()
    for neighbor in graph[current_node]:
        if visited[neighbor] == False:
            print(neighbor)
            deq.appendleft(neighbor)
            visited[neighbor] = True
```