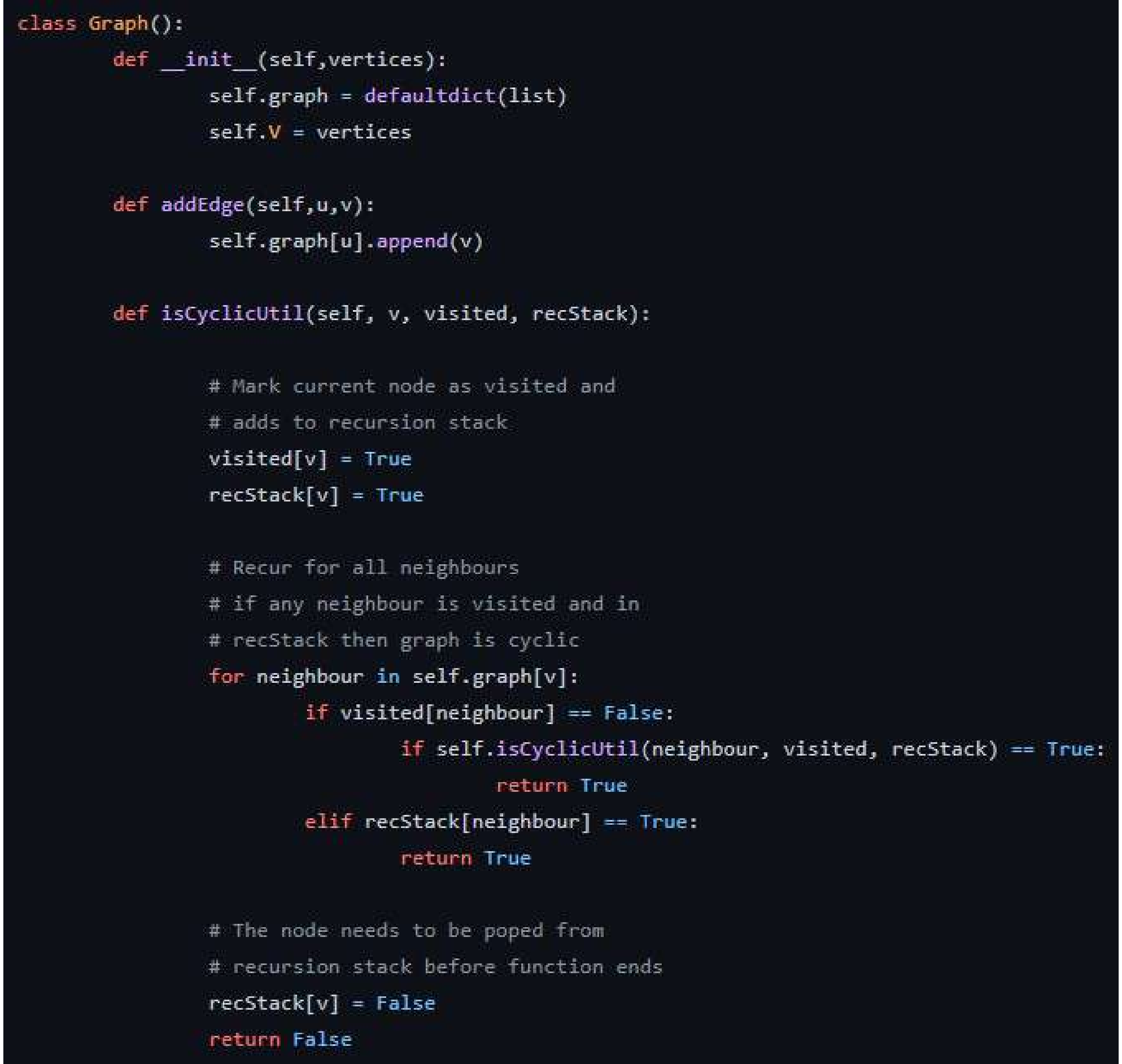
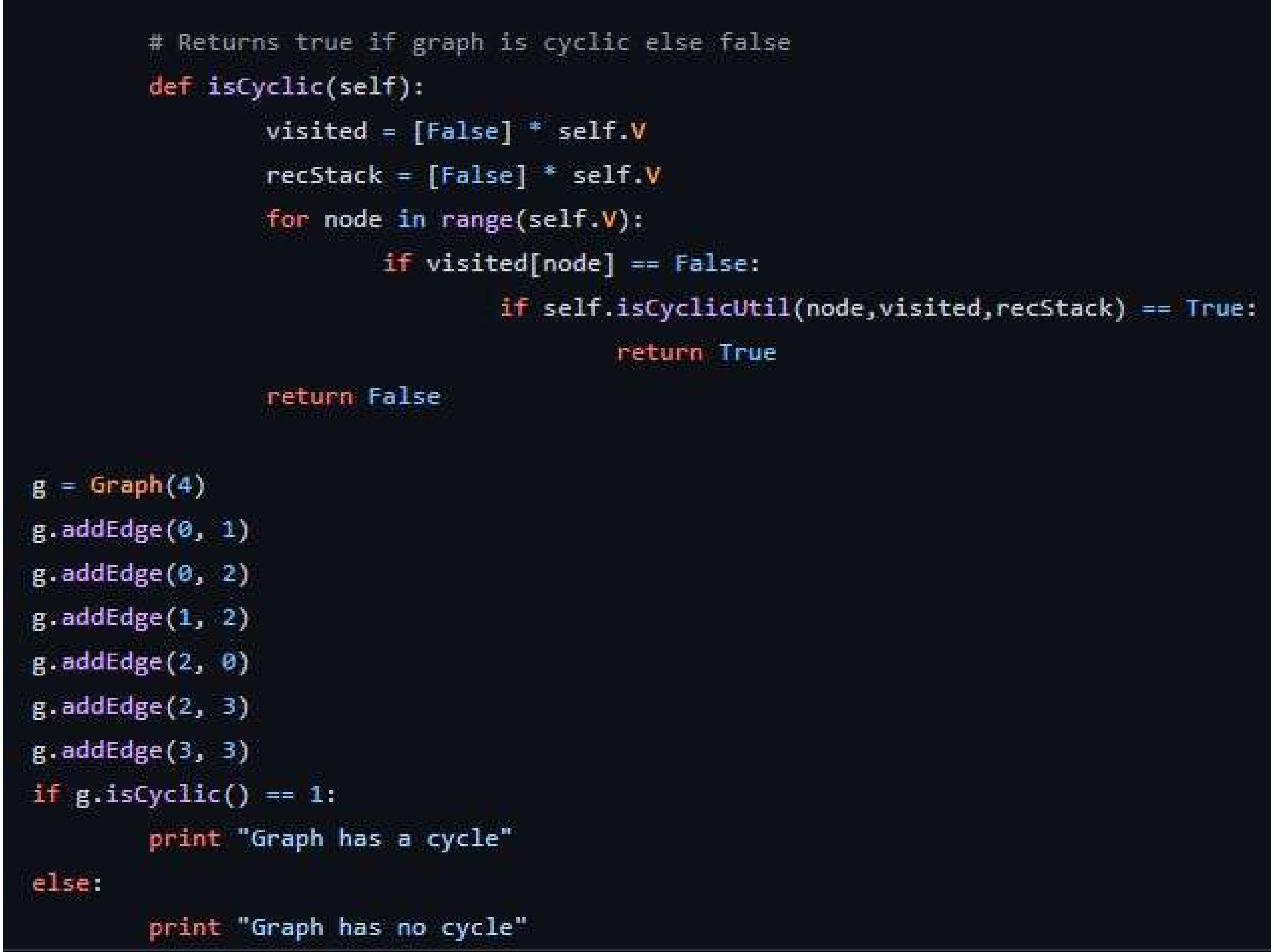


```python
class Graph():
        def __init__(self,vertices):
                self.graph = defaultdict(list)
                self.V = vertices

        def addEdge(self,u,v):
                self.graph[u].append(v)

        def isCyclicUtil(self, v, visited, recStack):

                # Mark current node as visited and
                # adds to recursion stack
                visited[v] = True
                recStack[v] = True

                # Recur for all neighbours
                # if any neighbour is visited and in
                # recStack then graph is cyclic
                for neighbour in self.graph[v]:
                        if visited[neighbour] == False:
                                if self.isCyclicUtil(neighbour, visited, recStack) == True:
                                        return True
                        elif recStack[neighbour] == True:
                                return True

                # The node needs to be poped from
                # recursion stack before function ends
                recStack[v] = False
                return False
```



```
    # Returns true if graph is cyclic else false
    def isCyclic(self):
        visited = [False] * self.V
        recStack = [False] * self.V
        for node in range(self.V):
            if visited[node] == False:
                if self.isCyclicUtil(node,visited,recStack) == True:
                    return True
        return False

g = Graph(4)
g.addEdge(0, 1)
g.addEdge(0, 2)
g.addEdge(1, 2)
g.addEdge(2, 0)
g.addEdge(2, 3)
g.addEdge(3, 3)
if g.isCyclic() == 1:
    print "Graph has a cycle"
else:
    print "Graph has no cycle"
```