In [6]:

```
class DirectedGraph:
    def __init__(self, vertices):
        self.vertices = vertices
        self.adjacency_list = {}
        for vertex in range(vertices):
            self.adjacency_list[vertex] = []
    def add_edge(self, u, v):
        self.adjacency_list[u].append(v)
    def is_loop(self, vertex, visited, stack):
        visited[vertex] = True
        stack[vertex] = True
        for neighbor in self.adjacency_list[vertex]:
            if not visited[neighbor]:
                if self.is_loop(neighbor, visited, stack):
                    return True
            elif stack[neighbor]:
                return True
        stack[vertex] = False
        return False
    def is_cyclic(self):
        visited = [False] * self.vertices
        stack = [False] * self.vertices
        for vertex in range(self.vertices):
            if not visited[vertex]:
                if self.is_loop(vertex, visited, stack):
                    return True
        return False
graph = DirectedGraph(6)
graph.add_edge(1,2)
graph.add_edge(2,4)
graph.add_edge(4,1)
graph.add_edge(3,2)
graph.add_edge(4,3)
graph.add_edge(3,2)
if graph.is_cyclic():
    print("The graph contains a loop")
else:
    print("The graph does not contain a loop")
```

```
The graph contains a loop
```