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

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

    def cycle(self, v, visited, stack):
        visited[v] = True
        stack[v] = True

        for i in self.graph[v]:
            if visited[i] == False:
                if self.cycle(i, visited, stack) == True:
                    return True
            elif stack[i] == True:
                return True
        stack[v] = False
        return False

    def isCycle(self):
        visited = [False] * (self.V + 1)
        stack = [False] * (self.V + 1)
        for node in range(self.V):
            if visited[node] == False:
                if self.cycle(node, visited, stack) == True:
                    return True
        return False

if __name__ == '__main__':
    g = Graph(4)
    g.addEdge(1,2)
    g.addEdge(2,4)
    g.addEdge(4,1)
    g.addEdge(4,3)
    g.addEdge(3,2)

    if g.isCycle() == 1:
        print("Graph contains cycle")
    else:
        print("Graph doesn't contain cycle")
```

```
Graph contains cycle
```