```
'''
    Time complexity: O(N^3)
    Space complexity: O(N^2)

    where N is the number of vertices in the input graph and
    M is the number of edges in the input graph

'''
from sys import stdin

def getCycles(graph, n):
    cycleCount = 0

    for i in range(n-2):
        for j in range(i+1,n-1):
            for k in range(j+1,n):
                if (graph[i][j] and graph[j][k] and graph[k][i]):
                    cycleCount+=1
    return cycleCount



n,m = list(map(int,stdin.readline().strip().split( )))
graph=[[ None for i in range(n) ] for j in range(n) ]

for i in range(m):
    u,v = list(map(int,stdin.readline().strip().split( )))
    graph[u][v] = True
    graph[v][u] = True


print(getCycles(graph, n))


'''
    An optimised solution can also be written
    whose complexity may be as follows:

    Time complexity: O(N^2)
    Space complexity: O(N + M)
    where N is the number of vertex in the graph
    and M is the number of edges in the graph
'''
```