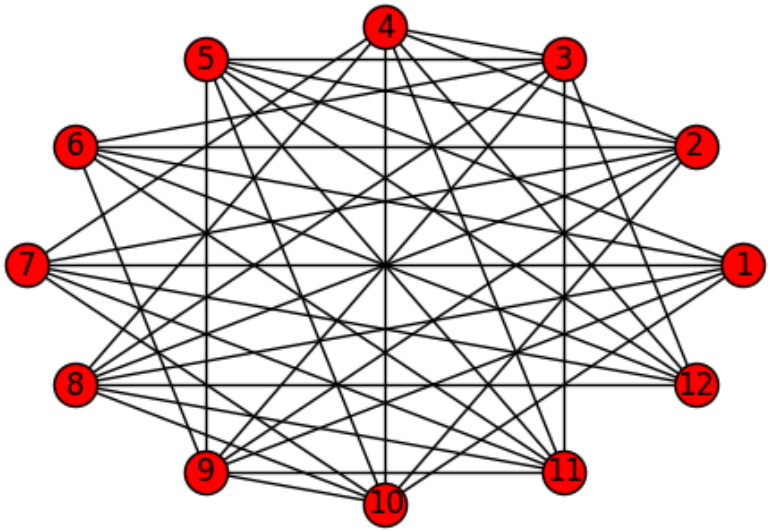
Challenge Problem 7

Jacob Craffey

In [14]:
```
import warnings
warnings.filterwarnings('ignore')

import networkx as nx
import matplotlib.pyplot as plt

def check_color(node, neighbor, node_color):

    # If the neighbor is less than the current node, then that edge has alread
y been checked and exits the function
    if (neighbor < node):
        return

    # Compares the two nodes' colors.  If they are the same, then the node rec
ieves a new color
    if (node_color.get(node) == node_color.get(neighbor)):
        node_color[neighbor] += 1

def coloring(graph):

    # Creates a dictionary that tracks the color of each node and assigns an i
nitial color of 0
    node_color = {}
    for node in graph:
        node_color[node] = 1

    # Goes through every edge in the graph and calls method check_colors
    for node in graph:
        for edge in graph[node]:
            check_color(node, edge, node_color)

    # Returns the value with the highest color recorded.
    maximum = max(node_color, key = node_color.get)
    return node_color[maximum]




graph = nx.Graph({1: [5,6,7,8,9,10],
                  2: [4,5,6,7,8,9,10],
                  3: [4,5,6,8,9,11,12],
                  4: [2,3,7,8,10,11,12],
                  5: [1,2,3,9,10,11,12],
                  6: [1,2,3,9,11,12],
                  7: [1,2,4,10,11,12],
                  8: [1,2,3,4,10,11,12],
                  9: [1,2,3,5,6,10,11],
                  10: [1,2,4,5,7,8,9],
                  11: [3,4,5,6,7,8,9],
                  12: [3,4,5,6,7,8]})

nx.draw_circular(graph, with_labels = True)
plt.show()
containers = coloring(graph)
print (containers)
```



4

Description of Graph

To begin, each lane has been represented as a node. If there is a chance where vehicles in two lanes collide with each other, they are connected with an edge.

I put the intersection in a NetworkX dictionary and is drawn in a circular pattern in order to present a rough visual representation of the intersection.

Strategy of Graph Model & Code

To begin, I created a seperate dictionary that holds the color of each corresponding node. Each node is then assigned the same initial color.

Then, the graph dictionary goes through a loop and each node and every edge passes through **check_color**. If the neighbor is less than the node, it skips that iteration since it has already been checked. The node and the neighboring node then compare colors. If they are the same, the color increments by one. Therefore, by the end of the loop, no node can share the same color with their neighbor.

After the loops has finished, we select the highest number that has been reached in our **node_color** dictionary. That is the lowest amount of colors needed for the graph coloring.

Discussion of Answer

We are given an intersection where we need to design the traffic stops which minimize traffic.

With a restricted path that the cars must take in their designated lanes, we can come to the conclusion that four light phases are needed to minimize traffic at this intersection.

To figure how this has been found, we first start with green lights for all lanes. Then we compare every lane with every other lane and check for possible collisions. While going through this process, if vehicles in lane *a* have a possibility to collide with vehicles in lane *b*, then there must be a light phase to prevent the collision.

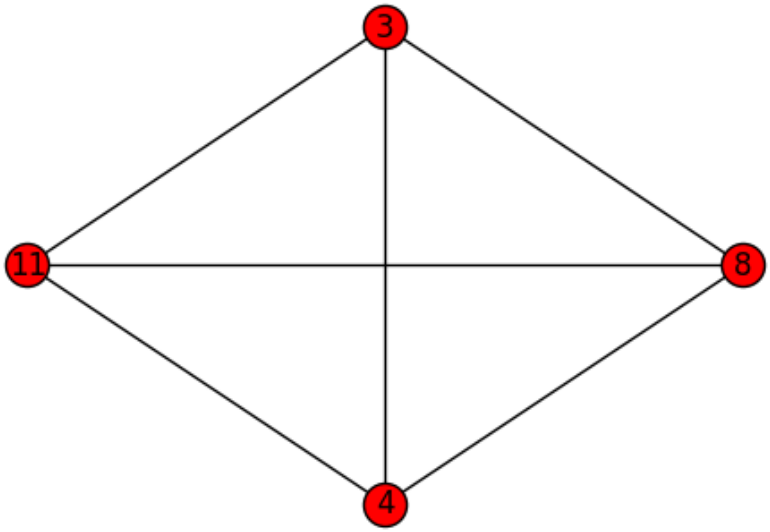
In order to minimize the amount of light phases needed for the intersection, lanes that are prone to collision are assigned to the pre-existing light phases. If they are still at risk of collision, the lane is re-assigned new phases until no collisions are possible.

Proof for Chromatic Coloring

To prove that the chromatic coloring for this graph is 4, I will give a specific situation in which a 3-coloring is not possible.

Here is a sub-graph from the graph above.

In [15]:
```
graph = nx.Graph({3: [4,8,11],
                  4: [3,8,11],
                  8: [3,4,11],
                  11: [3,4,8]})
nx.draw_circular(graph, with_labels = True)
plt.show()
```



Since this is a K_4, at least 4 colors are needed to complete this graph.