Challenge 3: Detect Cycle in a Directed Graph

Here's another coding challenge on directed graphs. You'll implement a cool function which detects loops!

We'll cover the following

Problem statement#

The concept of loops or cycles is very common in graph theory. A cycle exists when you traverse the directed graph and come upon a vertex that has already been visited.

You have to implement the `detect_cycle` function which tells you whether or not a graph contains a cycle.

Input#

A directed graph.

Output#

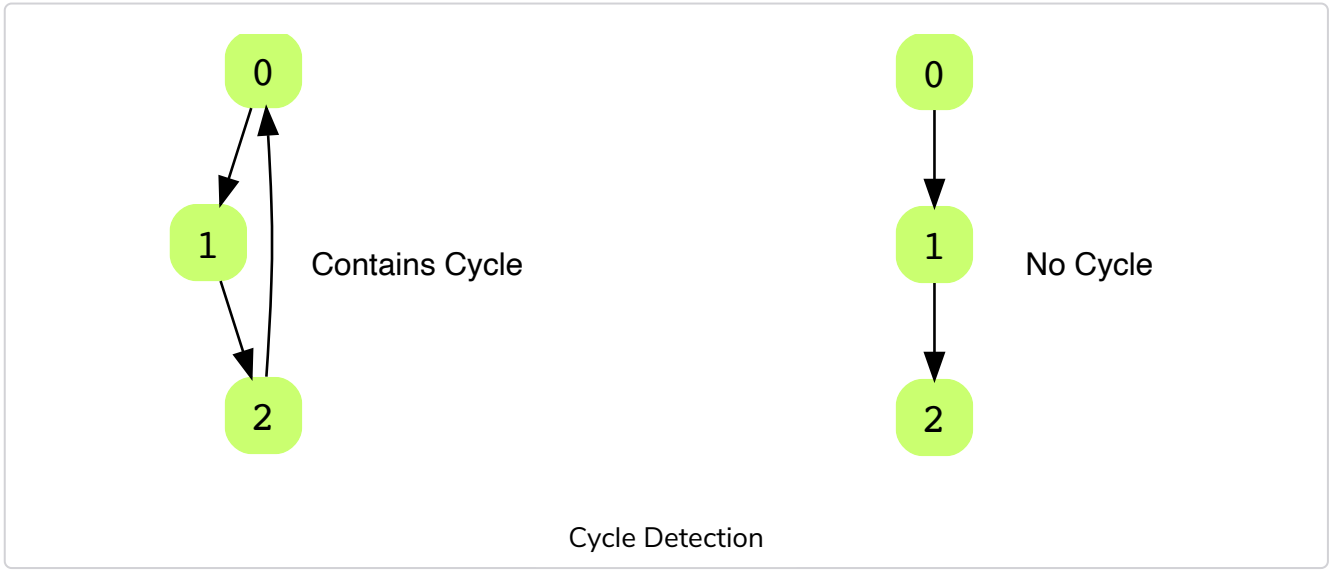
`True` if a cycle exists. `False` if it doesn't.

Sample input#

```
graph = {
    0 -> 1
    1 -> 2
    2 -> 0
}
```

Sample output#

```
True
```



Cycle Detection

Coding exercise#

Take a close look and design a step-by-step algorithm first before jumping on to the implementation. You can create as many helper functions as you nee

Try to think iteratively and recursively. If you get stuck, you [illegible] to the solution which we will discuss in the following lesson.

Good luck!

main.py

Graph.py

Stack.py

Queue.py

LinkedList.py

Node.py

```
from Graph import Graph
from Queue import MyQueue
from Stack import MyStack
# You can check the input graph in console tab

# Create Stack => stack = MyStack()
# Functions of Stack => push(int), pop(), top(), is_empty()
# Create Queue => queue = MyQueue()
# Functions of Queue => enqueue(int), dequeue(), size(), front(), is_empty()
# class Graph => {int vertices, linkedList[] array}
# class linkedList => {Node head_node}
# class Node => {int data, Node next_element}

def detect_cycle(g):
  # Write your code here
  pass

# Create any helper functions here
```

how

Back

Solution Review: Implement Depth Fir...

Next

Solution Review: Detect Cycle in a Dir...

Mark as Completed

Report an Issue