Challenge 3: Deletion by Value

Based on how we handled the deletion at head strategy, let's write the function for deletion by value.

We'll cover the following

Problem Statement

In this lesson, you'll be implementing the **delete by value** strategy. We'll describe its functionality, which should give you a clearer idea of what you have to do.

If you fully understood the last lesson, this should be a piece of cake.

In this function, we can pass a particular value that we want to delete from the list. The node containing this value could be anywhere in the list. It is also possible that such a node may not exist at all.

Therefore, we would have to traverse the whole list until we find the value which needs to be deleted. If the value doesn't exist, we do not need to do

anything.

Input

A linked list and an integer to be deleted.

Output

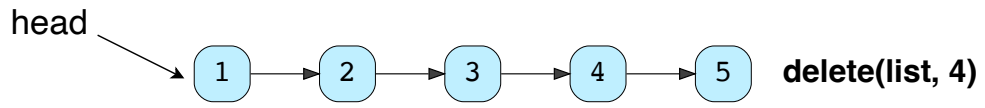
`True` if the value is deleted. Otherwise, `False`.

Sample Input

```
LinkedList = 3->2->1->0
Integer = 2
```

Sample Output

```
True
```



1 of 8

Coding Exercise#

You will now implement the `delete` function, which will take an integer value and delete the node containing it.

Based on what you've learned up till now, this exercise should be hard.

All the functions we've written, such as `search`, `is_empty`, and `delete_at_head` are available as members of the `LinkedList` class. To use any of these functions for a given `lst`, use

```
    lst.delete_at_head()
    lst.search()
    ...
```

The solution will be explained in the next lesson.

Good luck!

main.py

LinkedList.py

Node.py

```
from LinkedList import LinkedList
from Node import Node

# Access head_node => list.get_head()
# Check if list is empty => list.is_empty()
# Delete at head => list.delete_at_head()
# Search for element => list.search()
# Node class  { int data ; Node next_element;}


def delete(lst, value):
    # Write your code here
    if lst.is_empty():
       return False
    if lst.search(value) is False:
        return False
    else:
        prev = None
        current_node = lst.get_head()
        if current_node.data == value:
            current_node.delete_at_head()
```

```
            return True
        else:
            while current_node is not None:
                if current_node.data == value:
                    prev.next_element = current_node.next_element
                    return True
                prev = current_node
                current_node = current_node.next_element
    return False
```