Challenge 1: Insertion at Tail

Now, we'll tackle the second insertion strategy in linked lists. However, you will be the one implementing it!

We'll cover the following

Problem Statement

Just as heads and tails are polar opposites, this function is the opposite of what we saw in the last lesson. However, it is just as simple.

We need to insert a new object at the end of the linked list. You can naturally guess, that this newly added node will point to `None` as it is at the tail.

Input

A linked list and an integer value.

Output

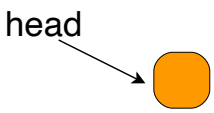
The updated linked list with the value inserted.

Sample Input

```
Linked List = 0->1->2
integer = 3
```

Sample Output

```
Linked List = 0->1->2->3
```



Insert 1

1 of 9

Coding Exercise

Now that we've understood the theory, it's time for you to write the practical implementation. In the code provided below, you must complete the `insert_at_tail()` function.

It will take an integer and a linked list as input parameters and insert a node containing that value at the end of the list.

You only have to write code where instructed in the code snippet below. There is no need to print anything or alter the `return` statement.

The `Node` and `LinkedList` classes implemented in the previous lessons are available to you.

Test your code against our cases and see if you can pass them.

There are hints and the solution in case you're stuck. We'll also be discussing the solution.

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()
# Node class  {int data ; Node next_element;}

# Inserts a value at the end of the list


def insert_at_tail(lst, value):
    # Write - Your - Code
    pass

```

Insertion At End

Back

Singly Linked List Insertion

Next

Solution Review: Insertion at Tail

Completed