Module 4 Lab: Linked List

Write a linked list that supports O(1) add_first, add_last, and remove_first operations.

Your starter code includes `Lab4.py` which has a complete `Node` class and a few methods in a `LinkedList` class (`iter`, `len`, and `repr`). It also includes `TestLab4.py` which has the full suite of test cases being run in Gradescope - feel free to run that locally to monitor your progress and to consult it to help clarify expected behavior for methods.

Part 1 - Constructor method with optional starting items

In python, default arguments for functions are only initialized once. This means we should not use a mutable argument as a default, since it can change over time:

```
>>> class Foo:
...     def __init__(self, L=[]): # Empty list is a BAD default
...             self.L = L
...
>>> x = Foo() # x.L is the default empty list
>>> y = Foo() # y.L is the SAME default list
>>> x.L.append(3)
>>> x.L
[3]
>>> y.L
[3]
>>>
```

If we want to make a custom collection with an optional collection of arguments, we should use an immutable like `None` for our default list, and create an empty list on the fly *inside* of our constructor method `init`:

```
>>> class Bar:
...     def __init__(self, L=None):
...             if L is None:
...                     self.L = [] # new empty list created for every object
...             else:
...                     self.L = L # whatever the user passed in
...
>>> x = Bar()
>>> y = Bar()
>>> x.L.append(3)
>>> x.L
[3]
>>> y.L
[]
```

Complete the constructor method for your LinkedList class such that it takes an optional parameter `items`. Use `None` as the default value. If the value of `items` is `None`, then create an empty LinkedList. Otherwise, the user has passed in some collection of items, and you should add them to your LinkedList one at a time using either `add_first` or `add_last`. Choose the method that lets you maintain order when possible:

```
>>> ll1 = LinkedList([0, 1, 2, 3, 4])
>>> print(ll1) # note the order is preserved below
LinkedList:
        Head: Node(0)
        Tail: Node(4)
        0-->1-->2-->3-->4-->None
```

You will also need to add parameters for tracking the head (`_head`), tail (`_tail`), and length (`_len`). You should name the parameters as given here, so the provided methods `len`, `iter`, and `repr` will work.

Part 2 - Adding and removing

Next, implement `add_first(item)`, `add_last(item)`, and `remove_first()`. All 3 methods should run in $\mathcal{O}(1)$.

Part 3 - Exceptions

`remove_first()` should raise a `RuntimeError` if called on an empty LinkedList.

Submitting

At a minimum, submit the following files:

- `Lab4.py`

Students must submit **individually** by the due date (typically, Sunday at 11:59 pm EST) to receive credit.