

Lab 5: Linked Lists Solution

4) Additional exercises

Generalizing `__getitem__`

The implementation we've provided for `__getitem__` has many shortcomings compared to Python's built-in lists.

Two features that it doesn't currently support are negative indexes and slices (e.g., `my_list[2:5]`).

Your first task here is to investigate the different ways in which Python supports these operations for built-in Python lists; you can do this by experimenting yourself in the Python console, or by doing some reading online.

Then, modify the linked list implementation of `__getitem__` so that it handles both negative indexes and slices.

Note that a slice in Python is actually a class: the expression `my_list[2:5]` is equivalent to `my_list.__getitem__(slice(2, 5))`.

Use `isinstance` to determine whether the input to `__getitem__` is an integer or a slice.

The fully general method signature of `__getitem__` should become:

```
1  def __getitem__(self, index: Union[int, slice]) -> Union[Any,
    LinkedList]
```

Note: slicing should always return a new `LinkedList` object.

This means that for a given slice, you'll need to create a `LinkedList` and new `Nodes` as well, in a similar manner to how you implemented the more powerful initializer at the end of Task 1.

Negative Index:

```
1  class LinkedList:
2      ...
3      def __getitem__(self, index: int) -> Any:
4          """Return the item at position <index> in this list.
5
6          Raise IndexError if <index> is >= the length of this list.
```

```

7
8     >>> lst = LinkedList([1, 2, 10, 200])
9     >>> lst[-1]
10    200
11     >>> lst[2]
12    10
13     >>> lst[-10]
14    Traceback (most recent call last):
15        ...
16    IndexError
17    """
18    curr = self._first
19    curr_index = 0
20    # ===== (Task 4, step 1) =====
21    index = index if index >= 0 else self._length + index
22
23    if index < 0:
24        raise IndexError
25    # =====
26
27    while curr is not None and curr_index < index:
28        curr = curr.next
29        curr_index += 1
30
31    assert curr is None or curr_index == index
32
33    if curr is None:
34        raise IndexError
35    else:
36        return curr.item

```

Matplotlib Practice

Use *matplotlib* to plot the results of your timing experiments, using the same approach as last week (See matplotlib section in lab 4).