CS 120: Intro to Computer Programming II

In-Class Activity - 03 Linked Lists - Day 5

Activity 1 - Turn in this one

Insertion has two special cases: the list is empty, or the new node goes at the head of the list. Add `if` statements, at the start of your code, which will handle those two cases.

Write both the `if` condition, and also the code to implement each one.

WARNING: The order of these two `if`'s matters, a lot! What happens if you get them backwards?

Solution:

```
if head is None:
    head = new_node
elif head.val > node_node.val:
    new_node.next = head
    head = new_node
```

Activity 2 - Turn in this one

Sometimes, when you perform an operation on a list, you need to handle special cases at the end as well. This is certainly true with insertion!

Think carefully about what would happen if the node you're inserting needs to get added to the **end** of the list. Would your code crash? How can you improve it to prevent this?

Solution: Our current code accesses `cur.next.val` in the `while` loop condition. But what if `cur` points to the end of the list?

We should add a guard, so that if `cur` is at the end of the list, then we will not check the `val` of `cur.next`.

Will will update the `while` condition:

```
while cur.next is not None and cur.next.val < new_node.val:
```

Challenge Activity - Do not turn in this one

Did you notice that it's actually possible to combine the two special cases in insertion (empty list, and new node goes at the head)? Combine the conditions for your two `if` statements, like this:

```
if thing1 or thing2:
```

And then see if you can come up with a single block of code (no more `if`'s allowed!) which will handle both cases.

Solution:

```
    if head is None or head.val > node_node.val:
        new_node.next = head
        head = new_node
```