CS 120: Intro to Computer Programming II

In-Class Activity - 03 Linked Lists - Day 3

Activity 1 - Turn in this one

In the slides, we looked at the "canonical" `while` loop for iterating over a linked list. This didn't have many features, but it showed the common elements that (almost) every loop over a linked list will need.

Work together in your group to recreate this loop. If possible, do it without reviewing the video, or the slides. But once you're done, double-check it by looking a the Linked List slide deck.

Activity 2 - Turn in this one

In video, after we looked at the "canonical loop," I asked you if that loop had a bug, when the list started out empty. Now it's time to figure that out!

Simulate the code by hand, carefully checking out what happens at each step. If `head` is `None` at the beginning, what happens? Does the code crash? Does it have some subtle bug? Or does it just work, somehow?

Turn in a clear explanation of what happens; you a combination of words and/or reference diagrams - whatever you think will make it clearer.

Activity 3 - Turn in this one

Now, let's elaborate on your basic linked list. Write a snippet of code which will iterate over a linked list, and print the value of **every other** node (starting with the head).

For this version, use a counter. That is, keep a counter of how many nodes you've seen so far, and then have an `if` statement, inside the body of the loop, that decides when to print, and when not to.

Activity 4 - Optional

OPTIONAL. Complete this if you have time, and turn it in. If you don't have time, you may report to your TA that you ran out of time.

Now, repeat the exercise, but instead of using a counter, you should advance the `cur` pointer **two steps** on each iteration of the loop.

But wait! There's a danger here. Is it possible to "fall off the list" without noticing it? That is, what if `cur` is not `None`, but the next node is?

Experiment with the following code, see what Python will do with it:

```
cur = None
cur = cur.next
```

Turn in **two things** from this Activity:

- An explanation of what Python does if you try to read the `next` field of a `None` object
- Your code for the "print-every-other" loop - but including some code to make sure that you never go past a `None` pointer!

(activity continues on the next page)

Challenge Activity - Do not turn in this one

In the slides, we've shown this example of a function which prints out all of the values in a linked list, and then also returns the length of the list to the caller:

```
def print_list(head):
    cur   = head
    count = 0

    while cur is not None:
        print(cur.val)
        count += 1
        cur = cur.next

    return count
```

Write a new function, `print_list_1()`, which does **exactly the same work** - but in a different way. In this new function, if the list is empty, then handle it here, in this function. Otherwise, handle only the **first** node of the list, and the **call** `print_list()` - passing it `head.next` - to handle the entire rest of the list.

Write the code for this function, paying attention to:

- What will `print_list()` print for you, and what will it return?
- You must print anything that `print_list()` doesn't print for you. What should you print, and when?
- Based on what `print_list()` returns, what should you return?