Assignment 1 Announcements

Extra office hours starting---check course website

Some updates for python_ta---check A1 handout

FAQ on the course forum!

Additional sample tests to be posted today.
Other announcements

Posted materials from last Friday! (Sorry for the delay)

**Midterm 1** on October 23\textsuperscript{rd}! We’ll post more information this afternoon.
Linked Lists

CSC148, INTRODUCTION TO COMPUTER SCIENCE
DAVID LIU
There are two major list implementations

**Array-based** lists store references to elements in contiguous blocks of memory.

**Linked** lists can store elements anywhere, but each element must store a reference to the *next* element in the list.
Our goals this week

1. Work with linked lists by implementing same operations as Python’s built-in list.
2. Analyze the running time of our linked list methods and compare them to the array-based list.
Code summary

class _Node:
    item: Any
    next: Optional[_Node]

class LinkedList:
    _first: Optional[_Node]

curr = self._first
while curr is not None:
    ... curr.item ...
    curr = curr.next
Takeaways

Code templates are useful.
Code templates aren’t everything.

Writing a stopping condition is often easier to understand than writing a loop condition.
Linked list insertion and deletion

It’s all about the links.
def insert(self, index: int, item: Any) -> None:
    """Insert the given item at the given index.
    Raise IndexError if index > len(self) or index < 0.
    Adding to the end of the list is okay.
    """
def pop(self, index: int) -> Any:
    """Remove and return the item at the given index.

    Raise IndexError if index >= len(self) or index < 0.
    """