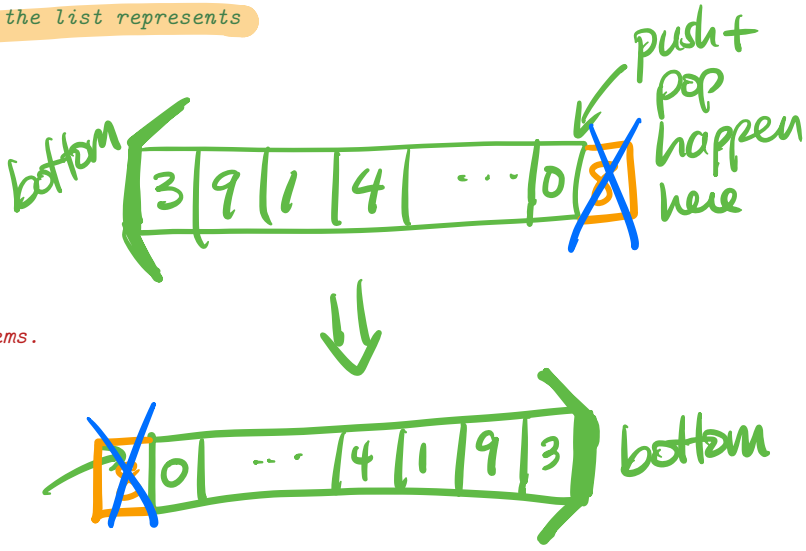
CSC148 - Considering a different implementation of class Stack

Below is the complete stack class that you saw in the readings.

```
class Stack:
    """A last-in-first-out (LIFO) stack of items.

    Stores data in first-in, last-out order. When removing an item from the
    stack, the most recently-added item is the one that is removed.
    """
    # === Private Attributes ===
    # _items:
    #     The items stored in the stack. The end of the list represents
    #     the top of the stack.
    _items: List

    def __init__(self) -> None:
        """Initialize a new empty stack.
        """
        self._items = []

    def is_empty(self) -> bool:
        """Return whether this stack contains no items.

        >>> s = Stack()
        >>> s.is_empty()
        True
        >>> s.push('hello')
        >>> s.is_empty()
        False
        """
        return self._items == []

    def push(self, item: Any) -> None:
        """Add a new element to the top of this stack.
        """
        self._items.append(item)

    def pop(self) -> Any:
        """Remove and return the element at the top of this stack.

        >>> s = Stack()
        >>> s.push('hello')
        >>> s.push('goodbye')
        >>> s.pop()
        'goodbye'
        """
        return self._items.pop()
```



1. We are going to revise the code so that instead of putting the top of the stack at the end of the list, the top of the stack is at the *beginning* of the list.

   (a) What code has to change?

done

   (b) Make the changes. done

2. What docstrings must change to go along with your code changes? Explain.

done

3. What changes would be required to the parenthesis balancer code that we worked on in the last lecture in order for it to work with this new version of the class? Explain.

None run it + see.

4. Which implementation do you think is better? What criteria are you using?

faster is better. Is one version of stack faster ??
≡ stability (easy to change without breaking)
less memory is better ≡
≡ less error prone → clean, ≡ elegant, simple