4.2 Traversing Linked Lists

HOW TO TRANSVERSE A LINKED LIST:

- How to write code that visits each element in a linked list one at a time, regardless of the linked lists length

THE CODE SEGMENT (transversal template):

1. Initialize the index i (*0 refers to the start of the list*)
2. Check if we've reached the end of the list
3. Do smth with the current element `my_list[i]`
4. Increment the index

**Transversing a linked list : the temporary variable now refers to a particular _node object **

```
curr = my_linked_list._first   # 1. Initialize curr to the start of the list)
while curr is not None:        # 2. curr is None if we're reached the end of the list.
    ... curr.item ...          # 3. Do something with the current *element* curr.item.
    curr = curr.next           # 4. "Increment" curr, setting it to refer to the next node.
```

```
    def to_list(self) -> list:
        """Return a (built-in) list that contains the same elements as this list.
        """
        items = []
        curr = self._first
        while curr is not None:
            items.append(curr.item)
            curr = curr.next

        return items
```

MAIN TEMPLATE:

```
curr = my_linked_list._first
while curr is not None:
    ... curr.item ...
    curr = curr.next
```