4.1 Introduction to Linked Lists

THE CONCEPT OF "LINKS":

- Python list often requires elements to be shifted back and forth:
    - This is because these elements are stored in slots in memory that are beside each other
    - Solution: an element plus a reference to the next element!

NODE: A single element in a list

- List of n elements -> n `_Node` instances

```python
class _Node:
    """A node in a linked list.

    Note that this is considered a "private class", one which is only meant
    to be used in this module by the LinkedList class, but not by client code.

    === Attributes ===
    item:
        The data stored in this node.
    next:
        The next node in the list, or None if there are no more nodes.
    """
    item: Any
    next: Optional[_Node]

    def __init__(self, item: Any) -> None:
        """Initialize a new node storing <item>, with no next node.
        """
        self.item = item
        self.next = None  # Initially pointing to nothing
```

A LINKEDLIST CLASS:

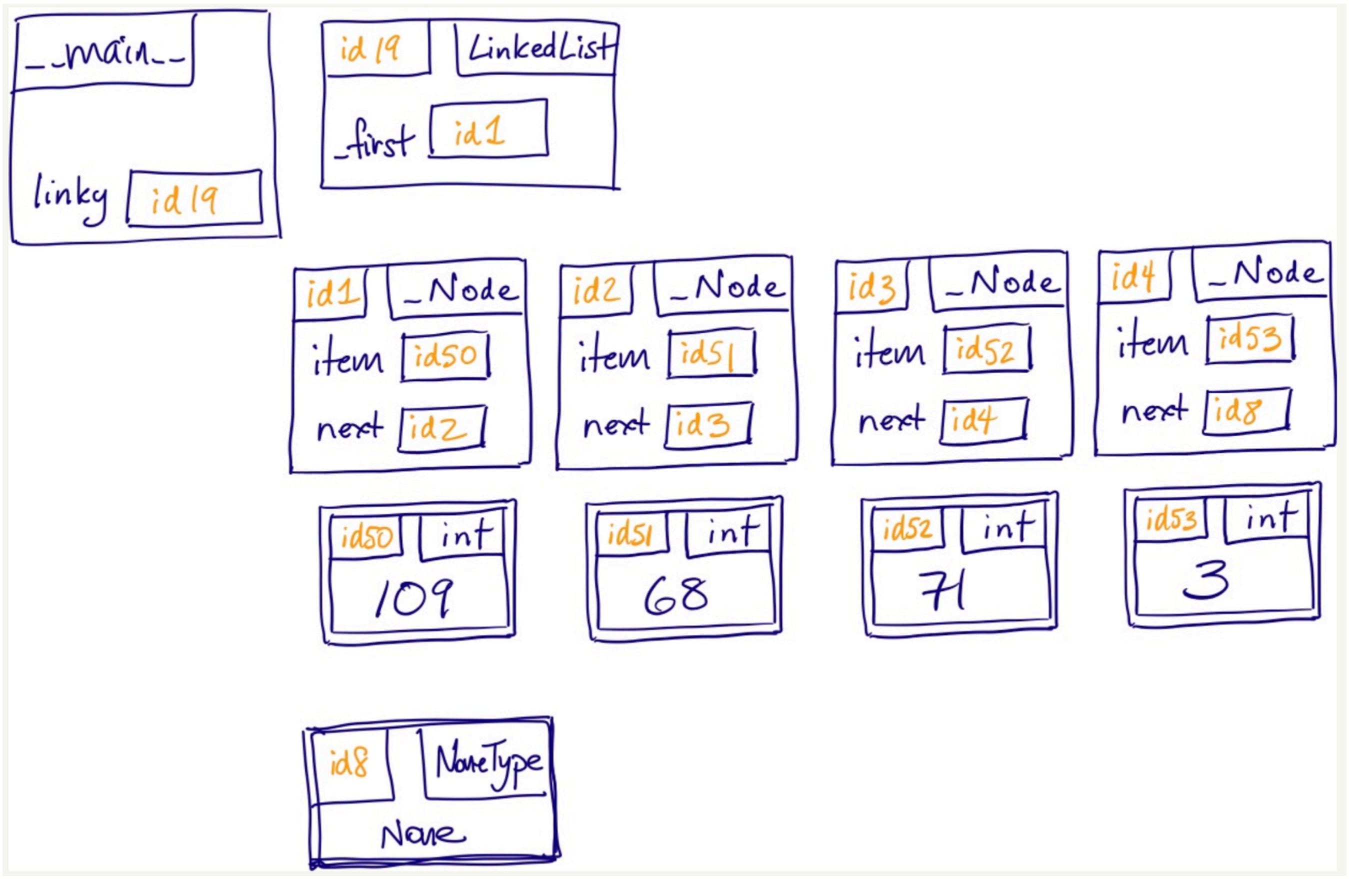
- List of elements
- ***Keep in mind: individual node object vs item it stores**
- Implementation

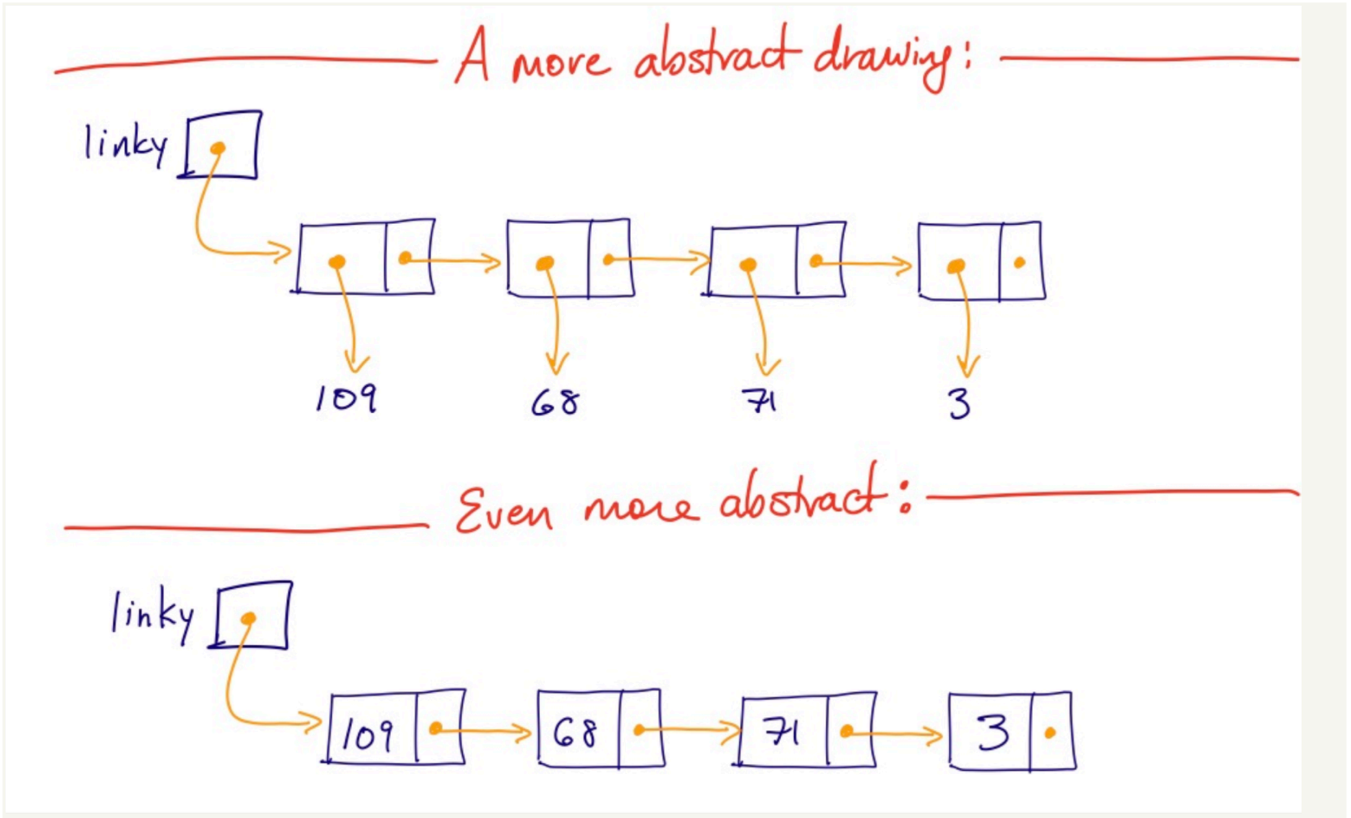
```python
class LinkedList:
    """A linked list implementation of the List ADT.
    """
    # === Private Attributes ===
    # The first node in this linked list, or None if this list is empty.
    _first: Optional[_Node]

    def __init__(self) -> None:
        """Initialize an empty linked list.
        """
        self._first = None
```

LINKED LIST DIAGRAMS:

- From super specific & time consuming -> to more straightforward





1. We use a preceding underscore for the class name to indicate that this entire class is *private*: it