Remove Element from Linked List

→ removing an element from the head of a singly linked list is essentially the reverse operation of inserting a new element at the head

```
remove_first(L):
  if L.head is None then
    error: list empty
  L.head = L.head.next      // make head point to next (or None)
  L.size -= 1               // decrement node count
```

Implementing a Stack using Singly Linked List

→ top of the stack at head: better choice because we can efficiently insert/delete elements in constant time only at the 'head'

```
class _Node:
  __slots__ = 'element', '_next'
  def __init__(self, el, next):
    self._el = el
    self._next = next

class LinkedStack:
  def __init__(self):
    self._head = None
    self._size = 0

  def __len__(self):
    return self._size

  def is_empty(self):
    return self._size == 0

  def push(self, e):
    self._head = self._Node(e, self._head)
    self._size += 1

  def top(self):
    if self.is_empty()
      raise Empty
    return self._head._element

  def pop(self):
    if self.is_empty()
      raise Empty
    answer = self._head._element
    self._head = self._head._next
    self._size -= 1
    return answer
```