Assignment 7: Hints

Here is some code to get you started in the current assignment.

1. The list nodes have an extra field, which must be created in the constructor. It should now read thus:

   ```
   def __init__(self, value = None, next = None, prev = None):
       self._value = value
       self._next = next
       self._prev = prev
   ```

   As a consequence, you must change the `__repr__` method in the `List_Node` class.

2. When you construct the `Double_List`, things are now very different:

   - the `_head` field is not `None` anymore
   - there is a `_tail` field
   - there are initially two sentinel nodes, which point to each other.

   Thus the `__init__` method should look like this:

   ```
   def __init__(self, orig = None):
       self._head = List_Node()       # create head sentinel
       self._tail = List_Node()       # create tail sentinel
       self._head._next = self._tail  # head points to tail
       self._tail._prev = self._head  # tail points to head

       if orig != None:
          for x in orig:
              self.append(x)
   ```

3. The `add_front` method has changed too. You should insert a node **after** the sentinel head node:

   ```
   def add_front(self, value):
       new_node = List_Node(value)       # create the node
       new_node._prev = self._head       # make it point back to sentinel
       new_node._next = self._head._next # make it point to old 1st node
       self._head._next = new_node       # sentinel now points to new
   node
       new_node._next._prev = new_node   # old 1st node points back to
   new node
   ```

   The `append` method is very similar.

4. The `__iter__` method walks down the list. It must be changed. You **don't** want to visit the sentinels. Only visit the data nodes.

```
def __iter__(self):
   # start at first DATA node (after head sentinel)
   current = self._head._next
   # keep going as long as you're not at the tail sentinel
   while current != self._tail:
       yield current._value
       current = current._next
```