Positional List ADT

→ serves as a more general abstraction; provides a way to refer to elements anywhere in a sequence, or perform arbitrary insertions & deletions

→ word processors often use a cursor to describe a position within a document w/o having to use integer indexes

→ using a linked list, we can perform O(1) time insertions & deletions, given a reference to a relevant node in the list

Positional List Abstract Data Type

→ a position instance is a simple obj, supporting only:

p.element(): return element stored at p

→ behaviors of a positional list

L.first(): return position of first element in L — L.before(p): return pos. of L immediately before p

L.last() — last element in L — L.after(p): — after p

iter(L): return a forward iterator for the elements of the list

L.add_first(e): insert e at front of L — L.add_before(p,e): insert e before p in L

L.add_last(e): — back of L — L.add_last(p,e): — after p in L

L.replace(p,e): replace element at p with element e

L.delete(p): remove and return element at p in L

→ the advantage of receiving a position means we can use it to navigate the list

example to print all elements of a positional list:

```
cursor = data.first()
while cursor is not None:
    print(cursor.element())
    cursor = data.after(cursor)
```