Sorting a Positional List

→ maintain a variable that represents the rightmost position of the currently sorted portion
→ during each pass, consider position just past marker as pivot and consider where the pivot's element belongs relative to the sorted portion
→ use another variable to move leftward from the marker as long as there is a preceding element with a value larger than the pivots

```
def insertion_sort(L):
  if len(L) > 1:
    marker = L.first()
    while marker != L.last():
      pivot = L.after(marker)          // next item to place
      value = pivot.element()
      if value > marker.element():     // pivot already sorted
        marker = pivot
      else:                            // relocate pivot
        walk = marker                  // find leftmost item greater than value
        while walk != L.first() and L.before(walk).element > value:
          walk = L.before(walk)
        L.delete(pivot)
        L.add_before(walk)             //reinsert value before walk
```

Maintaing Access Frequencies

→ consider maintaining a collection of elements while keeping track of the # of times each element is accessed
→ we model a new favorites list ADT with added methods:
  access(e): access element e, increasing count, add to list if not present
  remove(e): remove an element e from list if present
  top(k): return an iteration of k most accessed elements