Computer Science 134C
Introduction to Computer Science, in Python
Lecture #22 (Sorting)
November 5

Keywords
bubble sort, `__eq__`, insertion sort,
`__lt__`, ordered structures,
selection sort, `swap`

We review some simple ways search and sort.

1. Questions?
2. Loop invariants: Making statements about progress in loops.
3. How to compare values. How to swap them.
4. Bubble sort.
   (a) We can sort lists of items that are *totally ordered*: for any pair of values `a` and `b`, either `a <= b` or `b <= a`. Exactly when `a <= b <= a` we have `a == b`.
   (b) Suppose we are given a `list` of (totally ordered) values. Then we have `l[i] <= l[i+1]` or `l[i] > l[i+1]` for all `0 <= i < n-1`. Bubble sort simply makes passes through the array, comparing adjacent values, and swapping those out of order.
   (c) After each pass, we note the largest value encountered moves completely to the right.
5. Selection sort.
   (a) Like bubble sort, scan across the array, but keep track of the *location* of the largest value. After the scan, swap this value with the rightmost value.
   (b) As with bubble sort, the largest value encountered appears to the right.
6. Insertion sort.
   (a) Place a new value at the right of a *sorted* list. Now, push the value to its appropriate position to the left.
   (b) After each push, the list is fully sorted.
7. Quicksort.
   (a) Notice that we can identify where any value is correctly located within the sorted list, without actually sorting the list. For example, the smallest value should end up in location 0. We'll use a technique, called *partitioning* that finds the appropraite place for a *pivot* by putting smaller values to its left and larger values to its right.
   (b) Once partitioned, we can sort (somehow!) the smaller and larger values independently.
8. Merge sort. A favorite technique.
   (a) Split a list into two sublists; sort the sublists; merge them.

⋆