Sorting COMS 261

There are two built in ways to sort a list in Python.

`sorted()` function	`sort()` method
Works for any iterable type	Only works for lists
Returns a new list	Returns nothing (sorts in place)
Slightly slower	Slightly faster

1. What are the values of the following Python expressions? Use the shell to check your answers.
   (a) `sorted("cat")`

   (b) `sorted(range(4, 0, -1))`

   (c) `sorted((3, 1, 2))`

   (d) `"cat".sort()`

   (e) `[3, 4, 1, 2].sort() # Watch out, this one is tricky!`

2. The `sort` method and **`sorted`** function have an optional keyword argument called `key` that lets you choose a function to sort by. What do you get if you try the following command?

```
print(sorted(["Do", "a", "good", "job", "today"], key = len))
```

3. Both `sort` and **`sorted`** have another optional keyword argument called `reverse`. By default it is `False`. Let `scores = [87, 92, 74, 92, 60]`.
   (a) What does the following command do?

```
scores.sort(reverse = True)
```

   (b) Write a line of code that stores a copy of the scores from highest to lowest in a variable called `ranked`, without changing `scores`.

4. For each situation, decide whether you should use the `sort` method or the **`sorted`** function.

   (a) You have a list of names that you will never need in its original order, and you want to save memory.

   (b) You want to print the letters of a string in alphabetical order.

   (c) You are writing a function that receives a list as a parameter and returns the largest three values. The caller doesn't expect the list they passed in to change.

5. Python orders sequences of the same type **lexicographically**. Use this to answer the following.

   (a) What is the value of **`sorted`**`([("three", 3), ("two", 2), ("one", 1)])`?

   (b) What is the value of **`sorted`**`([(3, "three"), (2, "two"), (1, "one")])`?

6. Write a function `bottom_two(lst)` that returns the two lowest values in a list of numbers. You can assume there are at least two elements. The list `lst` should not be changed by your function.

```
def bottom_two(lst):
    """Returns a tuple with the two lowest values from lst."""
```

7. Write a function `is_sorted(lst)` that returns `True` if the elements of `lst` are in non-decreasing order and `False` otherwise. Do this *without* using `sort` or **`sorted`**.