CS 150: Intro to OOP, Spring 2009
Homework 1 Solutions

1. Exercises 2.6 and 2.7 from the text:

   (a) (3 points) Give a series of statements that determine and print the string that comes second, in alphabetical order, from an initially unsorted list.

   Solution: Assume that we have a list called 'mylist'. The following commands will print the second string in alphabetical order.

   ```
   mylist.sort()
   print mylist[1]
   ```

   ■

   (b) (2 points) Repeat part (a), ensuring that the original list remains unsorted.

   Solution: In order to find the second element without actually changing the list, we simply create a copy of 'mylist' to work with.

   ```
   copylist = list(mylist)
   copylist.sort()
   print copylist[1]
   ```

   ■

2. (2 points) Exercise 2.9 from the text:

   What is the result of the command `range`$(13, 40, 5)$?

   Solution: The result of the command `range`$(13, 40, 5)$ is:

   ```
   [13, 18, 23, 28, 33, 38]
   ```

   ■

3. (5 points) Exercise 2.11 from the text:

   Suppose that `numbers` is a list of values. Give a command or sequence of commands that outputs the median value of the list.

Solution: We may assume that the list numbers has already been created. The median value is the exact middle of the sorted list if the number of elements is odd. If the number of elements is even, you can either take the average of the middle two elements, or simply return one of the two middle elements. Either answer was perfectly acceptable. (Note that to take the average, you would need to use an if statement.)

Here is the code which simply returns a middle element.

```
numbers.sort()
middle = len(numbers)/2
print numbers[middle]
```

Here is the code which averages the two middle values if the length of the list is even.

```
numbers.sort()
if len(numbers) % 2 == 1:
   median = numbers[(len(numbers)-1)/2]
else:
   lowmiddle = (len(numbers)/2)-1
   highmiddle = len(numbers)/2
   median = (numbers[lowmiddle] + numbers[highmiddle])/2
print median
```

■

4. (4 points) Exercise 2.19 from the text:

   Write an expression that is **True** precisely when `stringA` is a capitalized version of `stringB`.

   Solution: This was a bit more complicated that it seemed. The function string.capitalize() changes the leading character to a capital letter, but also changes all the rest of the letters to lower case. So you need to be sure to check that the strings are identical after the first letter, and then that the first letters are upper and lower case versions of each other. You can't simply use .lower() for the second check, since that command makes the entire string lower case, and we just want to make sure the first letter of stringB has a lower case letter. The following code will give the desired result.

   Note: Any correct answer will be give full credit; you did not have to do it the same way that I did.

```
stringA[1:len(stringA)] == stringB[1:len(stringB)] and
stringA.lower()[0] == stringB[0] and
stringA[0] == (stringB.capitalize())[0]
```

■

5. (4 points) Exercise 2.22 from the text:

   Assume that `person` is a string in the form of 'firstName middleName lastName'. Give a command or series of commands that results in the creation of a corresponding string 'firstName middleInitial lastName'. For example, if `person` = 'Elmer Joseph Fudd', then the result should be 'Elmer J. Fudd'.

Solution: The following code will give the correct answer.

```
name = person.split()
print name[0], (name[1])[0] + '.', name[2]
```

■

6. (6 points, 1 per part) Exercise 2.27, parts b, c, e, i, k, and n

   Solution: (b) type: float, value = 6.5

   (c) type: int, value = 2

   (e) type: int, value = 26

   (i) type: str, value = 'ababab'

   (k) type: bool, value = True

   (n) type: int, value = 19

■