

Homework 2-3

Due: Oct. 13, 2011, 2:25 pm

Reminders

For the following problems you may discuss the concepts that will help solve these problems with classmates and course staff. You may *not* simply copy down the answers of your classmates as that is a violation of the collaboration policy. The one exception to this rule are those problems marked as Independent. You may discuss independent problems *with course staff only*.

Task 1:

Look at the following Python program (the green numbers are line references and not part of the program):

```
1 numOfPC = 10
2 numOfMac = 4
3 inventory = [numOfPC, numOfMac]
4
5 def addOne(num):
6     result = num + 1
7     return result
8
9 addOne(numOfMac)
10 numOfPC = addOne(numOfPC)
11 inventory[1] = addOne(inventory[1])
```

- Without actually running the program in Python, write down the values of the variables `numOfPC`, `numOfMac`, `inventory` *after* the program has executed lines 3, 9, 10, 11. (Hint: when in doubt, review the way we reasoned about programs in class and follow it religiously)
- Now, Open IDLE. Under the 'File' menu in the top left corner, click on 'New Window' (or simply press Ctrl+N on your keyboard). This should open up a new file as we saw in class. Type in the program.

- c. By inserting some `print` statements, modify the program so that when you execute it, you can see what are the values of the variables `numOfPC`, `numOfMac`, `inventory` after lines 3, 9, 10, 11.
- d. Save your program somewhere with your favorite name (you need to hand in this program). Then press F5 to run the program. Do the outputs match your prediction in (a)? If there is a difference, try to explain it.

For this assignment you will first figure out your answers in the Python shell. When you feel like you've arrived at the right answers write them out by hand or type them up and hand it in to the TAs.

Task 2:

In class, we gave a list of things that you should *not* do when writing a Python program (see the last slide from today's class). Remember the reason that we have these 'taboos' is that we can reason about programs much more easily if we restrict our way of writing them. This task is meant to give you a taste about how whimsical programs can get if you defy these rules.

- a. Danny, a naive Python programmer, wants to write a function that, given two variables, swaps their values. Here is what he wrote:

```
1 def swapBAD(a, b):
2     '''swap the values of two variables.'''
3     temp = a
4     a = b
5     b = temp
6     return
7
8 x = 10
9 y = 42
10 swap(x, y)
11 print x, y
```

What do you think are the values of `x` and `y` at the end of the program? If you are not sure about your answer, test it by running the program. Explain in a few words why the swap function succeeds or fails. You should be able to do it by just following our in-class reasoning.

- b. Danny then decides to write another function that swaps the first two elements in a list. Here is what he wrote:

```
1 def swapListBAD(input):
2     '''swap the first two items in a list'''
3     temp = input[0]
4     input[0] = input[1]
5     input[1] = temp
6     return
7
8 myList = [10, 42]
9 swapList(myList)
10 print myList
```

Run this program. Does the elements in the list get swapped? Pretty weird huh? Both programs Danny wrote violated the rule that says “never modify the arguments within your function definition”. And as you have seen, they display quite different behaviors. There of course is a reason, but it is too complex to explain here. (You do not need to hand in anything for part (b); just keep in mind the taboos when you write programs in the future).

Again, these are not actually bad programs (especially the second one – it does what it should), but they violate our class rules. And as a result, we cannot explain their behaviors and therefore we should not imitate them (for now).

Handin

Email your typed up homework and programs to cs0931tas@cs.brown.edu and title the file 'YOURNAME'hw1.txt — for example, GiliKligerhw1.txt. You can also hand in the written part (except programs) directly to TAs.