# Homework 2-2

*Mar. 6, 2012, 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:**

Create a folder on your desktop called 'HW2-2'. Download the starter program and the text of *Moby-Dick* from the course website and save them in this folder. Right click on the Python program and select 'Edit with IDLE'. **When you write your functions, remember to write down what your functions does, what the arguments mean, by commenting your code (remember #)**

a. The starter program contains a function called `printMD1000` that takes no arguments, and print out the first one thousand characters in *Moby-Dick*. The file can be located in example:

   `Z:/WinData/Desktop/HW2-2/MobyDick.txt`.

   Look at it and make sure you understand why it should work. Then press `F5` to run it. You should see another IDLE window comes up. Then after the prompt type `printMD1000()` to **call** your function. Make sure it produces the correct answer(you do not have to hand in anything for this part.)

b. Now, write another function called `print1000` which takes one argument. Suppose in the folder 'HW2-2', I have bunch of text files: `MobyDick.txt`, `Hamlet.txt`, `Bible.txt`, etc (we will be providing `MobyDick.txt`, download this from the website). You want the argument to this function tells the function which text file to print. For example, if after running the program, calling `print1000('MobyDick.txt')` should print the first one thousand characters of the file that can be located in:

   `Z:/WinData/Desktop/HW2-2/MobyDick.txt`

and calling `print1000('Hamlet.txt')` should print the first one thousand characters of the file

`Z:/WinData/Desktop/HW2-2/Hamlet.txt`

assuming the file does exist. **Hint**: remember you can use the + operator to 'glue' strings together.

c. Now you're going to text the function you just wrote. Provide one example of a call to `print1000` (different from the ones we've provided) that works as expected.

d. Write a third function, `printN`, that takes two arguments. The first one indicates which file to print (same as the last function) and the second is a number indicating how many characters from the beginning to print. For example, calling `printN('MobyDick.txt', 500)` should print out the first five hundred characters in the file located at

`Z:/WinData/Desktop/HW2-2/MobyDick.txt`

e. Provide two examples of calls to `printN` that show that it works as expected. What are you looking for when you check that printN is correct?

When you're done, name your program `'YOURNAME'mobydick.py` — for example, `GiliKligermobydick.py`.

**Task 2:**

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])
```

a. 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). Write down these variables in a text file called `'YOURNAME'hw2-2.txt` — for example, `GiliKligerhw1.txt`.

b. 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 and name it `'YOURNAME'hw2-2Task2.py` — for example, `GiliKligerhw2-2Task2.py`. (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. Write this solution in your text file.

**Task 3:**

In class, we gave a list of things that you should *not* do when writing a Python program (see the last slide from Tuesday, February 28th'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 swapBAD(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.

Write your solutions in the same text file as Task 1.

Carl Friedrich Gauss (1777-1855) is one of the greatest mathematician of all time who contributed significantly to many fields in mathematics, statistics, physics and astronomy. There is a famous anecdote about him (whether true or not). When he was a little boy and was attending some elementary school in the countryside, his math teacher one day asked the class to add up all the integers from 1 through 100. The teacher thought it would take the kids a long while and he just got himself some time for a cigarette break. However, to his annoyance, little Carl raised his hand right after he pulled out a match.

'The answer is 5050.' said Gauss. The teacher frowned suspicously and sneered 'You are wrong. Do it again.' Honestly he didn't know the answer himself and there is no way to come up with the right answer *that* quickly. No, not with a computer, said the teacher to himself.

But Gauss looked down at his sketch paper and lifted up his head in about five seconds. 'I just checked my calculation. I'm sure 5050 is the right answer.'

'Now you're just messing with me.' said the teacher, as he strode through many amazed eyes towards Gauss' desk. 'Show me your work.'

'Well, if you add 1 and 100, you get 101. Adding 2 and 99 also gives 101. The same goes for 3 and 98, 4 and 97, etc. There are 50 such pairs from 1 to 100. So the answer is 101 times 50 which is 5050.'

**Task 4:**

a. We are not as smart as Gauss was so let's add 1 through 100 in the conventional way. Luckily we have computers and Python. Write a function that adds up all the numbers from 1 to `arg`, where `arg` is the argument to the function (**Hint**: Use iteration! To use it, you want to create a list containing integers from 1 through 100. There is a built-in function `range(x,y)` that does this. Try to call this function in the interactive environment to see what it does). Run it with 100 and see if it produces Gauss' answer. What about 1,000,000? We've provided some starter code that you can build on (HW2-2.py).

b. Next, write a function that adds up all the *odd* numbers from 1 to `arg`. (Hint: Iterate as you did in part (a), but add the number to your total only if it is an odd one. Use the function we've provided called `isOdd` to help.)

**Task 5:**

a. Write a function that determines if an integer `a` is in a list of integers `myList`. To do this, iterate through all elements in the list, and check if it is the same as the integer `a`. Once you see a match, return `True`. After the iteration, return `False` (why?). Be careful with the indentations!

b. **(Extra Credit)** Now write a function that determines if all of the integers in a list `myList1` are in another list of integers `myList2`. To do this, iterate through all elements in `myList1`, and check if each element is in `myList2` (hmm... sounds like a familiar task – maybe you can reuse the function you wrote for part (a)). Again, just fill in the corresponding function skeletons in HW2-2.py.

### Handin

Email your typed up homework and programs to `cs0931handin@cs.brown.edu`, with the subject line `'YOURNAME'HW2-2`. You should have 3 files:

a. `'YOURNAME'mobydick.py` — for example, `GiliKligermobydick.py`, for Task 1.

b. `'YOURNAME'hw2-2Task2.py` — for example, `GiliKligerhw2-2Task2.py`, for Task 2.

c. `'YOURNAME'hw2-2.txt` — for example, `GiliKligerhw1.txt`, for Task 2 and Task 3.

d. `'YOURNAME'hw2-2.py` — for example, `GiliKligerhw2-2.py`, for Task 4 and Task 5