INTRODUCTION TO COMPUTER SCIENCE I
Fall 2014
LAB 1
Input, output, and calculations

In this lab, you'll start programming in Python. You'll find out how to write some code, make it run, and submit your work.

1 Getting started

For today's lab, sit at any machine in the Seeley Mudd 014 or 007, or pull out your laptop. To get started, follow the instructions for your situation:

- **In 014 at a workstation:** Press three keys, CTRL, ALT, and DELETE, all at the same time, to get a login screen. Login with your Amherst College username and password.[1] Once you're logged in, click on the icon at the bottom left and search for the program IDLE. Choose IDLE 3.3 from the search list.
- **On your laptop:** Alternatively, if you want to work on your own computer, go to the Python download page in order to get Python and IDLE (they come together) and install them. **Be sure to download Python 3, not Python 2**, since there are significant differences. Once you've installed it, find the installed IDLE 3 application and run it.
- **In 007 at a workstation:** Finally, if you are sitting at one of the computers in Seeley Mudd 007, then ask the TA (or look at the whiteboard) for instructions on how to login to `remus` or `romulus`, two IT Linux servers, and then to bring up IDLE 3 there.

1.1 Trying a few statements

A window labeled *Python 3.3.2 Shell* should pop up. From now on, we'll call this the *shell window*.[2] IDLE provides a simple facility for creating Python programs. You can begin by typing Python statements into the shell window. For example, you might try the following, omitting the `>>>` prompt that IDLE supplies:

```
>>> 3+4
>>> i = 3+4
>>> print(i)
>>> course = "Computer Science 111"
>>> print ("Welcome to " + course + "!")
```

[1] If you aren't an Amherst College student and haven't already been in touch with our IT department about getting accounts on our system, please talk to us right away.

[2] The exact version number may differ, but *Python 3.x.y*, for any values of *x* and *y*, should be just fine.

Let's consider these statements. In the first statement, the addition is performed and its result shown, but the result is not otherwise stored anywhere for later use. In contrast, the second statement stores the result in a variable—the space named `i`—and then that result is displayed by the third statement.

The fourth statement shows a new type of data: a *string*, which is a sequence of characters. A variable may name a space that stores a string (as shown in the fifth statement). Moreover, we can use the + operator *concatenate* strings and them.

1.2 Taking input from the user

Sometimes, a program wants to prompt the user, and then have that user type in a value. For that purpose, we can use the Python function, `input()`. However, we must be careful with how this function works. Try the following statments, noting that each use of `input()` will cause Python to pause and wait for you to type something in. For each of the first two statements, type some simple numeric values (e.g., 5 and 7), and see if you can figure out what happens and why:

```
>>> x = input()
>>> y = input()
>>> print(x+y)
```

What result do you get? Is it what you expected? How could you get the result that you did expect? (Hint: Ask a professor or TA, because there's something new that we must tell you about.)

2 Creating a `.py` file

You can create and reuse a program in the following way. Within the shell window, choose the File menu item and then select "New Window". Type the following simple program into it:

```
print ("Hello world!")
```

Choose "Save" from the File menu, and save the program into a file called "hello.py". You likely want to create this file in a newly created `COSC-111` directory, since you will be making many such files for many labs and projects this semester. If you don't know how to make a new folder, ask your local neighborhood professor or TA for help. Once you've saved the file, go to the "Run" menu and choose "Run Module". If you've done everything right, your shell window should reappear. Within the shell window, you should see the output of your program, the words "Hello world!"

3 Downloading a program

Often your lab work will involve modifying a short program that we give you. Here's an example.

Use a browser to visitf `http://app.cs.amherst.edu/sfkaplan/2014/fall/COSC-111/lab-1` It will download a bit of Python code for you to see in your browser. Save what you see, placing the file into your `COSC-111` directory, naming the file `lab-1.py`

Now that you've saved the file, go to your shell window and open the file. What you see if the simple beginnings of a program, with some *comments*—lines that begin with the # symbol and contain notes to the programmer[3]—that suggest how you should modify and expand that program. You can first, however, run the program and see it the Python interpreter carry out all of the lines in quick succession.

3.1 Your task: Adding the arithmetic

Modify the `lab-1.py` program to read in, from the user, three integer values, namely: `a`; `b`; and `c`. It is your task to then compute three new values, each of which depends on some subset of `a`, `b`, and `c`. Specifically, you must compute `x`, `y`, and `z` as follows:

$$x = (b \bmod a) + 12$$
$$y = \frac{b}{a}$$
$$z = \frac{ab}{10c} - 1$$

But wait! There are two things you need to know about these calculations before you begin writing code for them. First, note the request for performing a *mod* calcuation for x. This operation could also be defined as the *integer remainder* operation. For example, given $75 \bmod 30$, given that 30 goes into 70 twice and then leaves a remainder of 15, we say that $75 \bmod 30 = 15$.

Second, when performing division, notice that you may introduce *fractional* or *floating point* numbers into the picture. For example, $\frac{11}{3}$ normally yields $3.66666\ldots$ Indeed, if you type the following statement into IDLE, you will see the given result:

```
>>> 11 / 3
3.6666666666666665
```

However, for the computations above, **we don't want the fractional results**. Instead, we want to perform *integer arithmetic*, where any fractional values are simply removed. Specifically, there is a special operator, `//`, that does exactly that:

```
>>> 11 // 3
3
```

So, be careful to use that particular form of division when needed in writing code for the formulae given above.

So what are these wacky arithmetic operations for? These will serve as your "magic decoder ring" for the wild goose chase, below ...[4]

Notice that the final part of the program prints the values of variables `x`, `y`, and `z` to the shell window. Therefore, the code that you add to the program must **create** and **assign** these variables their correct values, as described above.

[3] That's you!

[4] Put differently, these arithmetic operations were chosen so that, if you find the right values for a, b, and c, you will then calculate the useful-but-only-superficially-meaningful x, y, and z to find something.

3.2 Testing your program

Once you have added the lines of code that perform the strange arithmetic, you should test that your program works! Specifically, these are three arithmetic operations that you could perform with pen and paper or, for those so inclined, with a calculator.

Armed with a few *test cases*, now run your program with IDLE. When prompted by your program to enter values for `a`, `b`, and `c`, choose any one of your pre-determined trio of values for those variables. Then examine your program's output. Did it produce the values for `x`, `y`, and `z` that you expected? If not, then either your program or your test case contains an error, and you must determine which is at fault and fix it. If the output **does** match your expectation, then you have one (more) test case to support your belief that your program is correct.[5] Once your program has passed enough test cases to convince you that it is likely to be working correctly, then you should move on to ...

4 The wild goose chase, take I

Have you even been to the gym? Have you noticed, on the walls surrounding the main, old basketball court (**not** LeFrak), the pictures of so many alumni who have competed on various teams, going back over 100 years? Your mission, should you choose to accept it,[6] is to find **one particular person in one particular such photograph.**[7]

4.1 Finding the inputs

To find this photograph and the person in it, you must find three very important numbers. Finding them will require a bit of patience, frighteningly little ingenuity, and, one hopes, a sunny disposition. Here are the clues—none too subtle—for finding those three numbers:

`a`: This value is *the numeric portion of the street address of the Folger Shakespeare Library*. Truly low cunning is required to discover this value. Should you require more than two minutes for this task, hang your head in shame and avoid eye contact. *Bonus point:* Why might I have involved the poor Folger in this fiasco?

`b`: In the *Olds Mathematics Library/Reading Room*,[8] there is an old dictionary sitting upon a reading pedestal. You need to find the number of the page on which the word "fantod" is defined in this *Webster's Third New International Dictionary*.[9]

`c`: *Something there is that doesn't love a wall.*
Which class year of Amherst alumns donated a statue of the author of this quote to the college?

[5] Do not confuse this belief as being **proof** that your program is correct. Proving that program produces correct output in all cases is exceedingly difficult, and way outside of the scope of this course.

[6] I highly recommend that you **do** accept it.

[7] No, we are not kidding.

[8] Don't know where that is? Use The Google, Luke.

[9] Yup, another **bonus opportunity**: What author would get the howling fantods if he were alive to see my mimicry of his heavy use of footnotes?

4.2 Using the outputs

This next step should not surprise you. Go run your `lab1` program, and enter the values of `a`, `b`, and `c` that you worked so hard to obtain. From it, you will, of course, obtain values for `x`, `y`, and `z`. If you **round the values for the results**, you can then use them to find the person among the pictures of alumni athletes in the gym. To whit, follow these steps to find the person in question:

1. Go to the gym. Go to the *hallway on the north side of the basketball court*.[10] Then, look at the *north wall of that hallway*—that is, with your back to the basketball court itself.
2. Starting from the far left side of this wall, find the x^{th} column of photographs.
3. From the bottom of that column, find the photograph in the y^{th} row. **Note the team and year of this photograph.**
4. Within the photograph, find the middle row of people. On its far left is the (rather aged, at that time) head coach.
5. From the left side of that row of people, find the z^{th} person. **Note the exact name given for this person in the photograph.**

Recording your big find: At the top of your `lab1.py` file, add a single line that is a *comment* (that is, it begins with the `#` sign, making the interpreter ignore it) that specifies the two pieces of information that you noted from alumni photograph: the team/year of the photograph, and name of the person. If you have figured out any of the above *bonus points*, add that information to this file of answers. **Non-bonus-curiosity point:** There are at least two other photographs in that gym in which this same person appears. Enter into your collection of answers the year and sport of those two photographs.

When you have entered all of the information that you can gather, save your file.

5 Submitting your work

You'll submit all of your work electronically this semester. To do this, use a browser to visit:

```
https://www.cs.amherst.edu/submit
```

Login, choose our class from the list of classes, and then choose "Lab 1". This will take you to the submission page for today's assignment. Click the "Browse" button, and choose your "lab1.py" file. Click "Upload" and the file will be sent to us.

As the semester proceeds, you may find that you want to change your program after you submit it. That's fine. You can submit each assignment multiple times if necessary. We'll get the different versions and will know when each was submitted.

You can review your submission after you've submitted it by clicking on the appropriate link on the submission page.

[10] Don't know which way is north? Seriously, you can't figure that out?

6 Finishing up

When you're done working, close all of your windows, and then click on the icon in the lower left corner of the screen and choose "Log off".