

CS 1110, LAB 02: FUNCTIONS AND MODULES — SOME “HF”-LIGHTS

<http://www.cs.cornell.edu/courses/cs1110/2018sp/labs/lab02/lab02.pdf>

First Name: _____ Last Name: _____ NetID: _____

Getting Credit: **As always, strive to finish during the lab session** — it’s the best way to stay on track in this course.¹

1. CHECK THAT YOU’VE GOT THE RIGHT PYTHON

Open a command shell.² Start up Python interactive mode.³ At the Python interactive prompt, `>>>`, enter this:

```
2/3
```

Circle which result you get:

0 0.6666666666666666 something else

If you didn’t circle the middle answer, get help from a staff member immediately — you’ve got the wrong Python version, and we’ll help you fix this.




2. BUILT-IN OPERATORS AND FUNCTIONS WE’LL USE IN THIS LAB

2.1. Repeated string concatenation. You know that `*` is the multiplication operator for ints and floats. It turns out that `*` can also be used to “repeat” a string, in a sense that the following examples will clarify. Enter (or copy-paste) each of the following expressions into Python and write down the evaluation results (if any).

Lab authors: D. Gries, L. Lee, S. Marschner, W. White

¹But if you don’t manage finish during lab, here are the alternate checkoff opportunities: (a) at ACCEL Green room consulting hours, listed at <http://www.cs.cornell.edu/courses/cs1110/2018sp/about/staff.php> , **from today until Tue Feb 13 inclusive**, (b) at non-professorial TA office hours **from today to Wed Feb 14 3:45pm inclusive**, although at TA office hours, questions about course material or assignments take precedence over lab check-offs; or (c) during the **first 10 minutes of your next scheduled lab (Tue Feb 13 or Wed Feb 14)**. Beyond that time, the staff have been instructed not to give you credit.

Labs are graded on effort, not correctness. We just want to see that you tried all the exercises, and to clarify any misunderstandings or questions you have.

²*Windows 7*: “Command Prompt” in  **Accessories** in the Start Menu. *Windows 8 or 10*: just search for “Command Prompt” in the search box at the bottom of the Start Menu. *Mac*: “Terminal” in  **Applications**•**Utilities**, or type  + to call up Spotlight Search, and then type **Terminal** in the Spotlight Search window to find it.

³That is, enter “python” at the command-shell prompt.

Expression	Evaluation Result
<code>s = 'abc.'</code>	
<code>s</code>	
<code>s*2</code>	
<code>s*4</code>	
<code>n = 3</code>	
<code>s*n</code>	

2.2. Random number generation. The built-in module `random` contains functions for generating random numbers. One such function is `randint(a,b)`, which, if `a` and `b` are ints with $b \geq a$, returns a random integer drawn from between `a` and `b` inclusive.

Let's try it out. First, in Python, make the name of the `random` module accessible to Python by entering `import random` at the interactive prompt.

Next, circle the function call that is correct given the `import random` command you just entered.

`random.randint(1,3)`

`randint(1,3)`

Why would the other option would give you an error in this case?

Check your answer by entering it four separate times in Python. (Hint: minimize the chance of typos by using the up-arrow (`↑`) key to bring up the previous line you entered.) Write down the four numbers you get:

Now, **exit Python** by entering “`exit()`” or hitting `ctrl`+`Z` then `return` (Windows) or `ctrl`+`D` (Mac) at the Python interactive prompt (`>>>`). But don't close the command shell; you'll need it in Section 3.2.

3. NAVIGATING FOLDER (DIRECTORY) STRUCTURE IN A COMMAND SHELL

3.1. Put the lab files in a new folder `Desktop ▶ lab02`.

Create a new folder `lab02` on your Desktop.

For each lab 02 file listed at <http://www.cs.cornell.edu/courses/cs1110/2018sp/labs> , click⁴ on the file's name to download and save it to your newly created folder `Desktop` ▶ `lab02`.

3.2. Navigate in the command shell to the new folder. From Section 2.2, you should still have a command shell open but have exited Python interactive mode, so you should **not** see the “>>>” prompt.

At any given time, the command shell is “looking at” a specific folder, which we call the *active* or *working directory*. When the command shell starts, the working directory is your *home directory*. If you are using your own computer, this is probably the folder with your username on it; on the lab machines, it's `C:\Users\cit-labs` .

Next, open up a File Explorer (Windows) and Finder (Mac) window for your home directory, and put it next to the command shell.

Then, in the command shell, enter the command that correspond to your operating system:

Windows	some notes	Mac OS	some notes
<code>dir</code>	(stands for “ <u>d</u> irectory”)	<code>ls -l</code>	(that's lower-case “LS -L” — there's no numeral 1 there. The name comes from “ <u>l</u> ist directory contents”, and the -1 (“minus ell”) means to generate <u>l</u> ong output)

Describe what you see in the command shell and how that resembles what you see in the File Explorer (Windows)/Finder (Mac) window you put next to the command shell. (If you get some sort of error, ask for help.)

Now let us move the view of the command shell to the Desktop. This is actually a folder inside your home directory. To get the command shell there, enter into the command shell (on either OS)

`cd Desktop`

(“`cd`” stands for change directory.) Your prompt should say “Desktop” in it somewhere; flag down a staff member if not.

Once again enter either `dir` or `ls -l` (depending on your operating system) into the command shell. Is the command shell still listing the files that are in your home directory, or is it now displaying the files that are on your Desktop, or something else? (If you get an error, ask for help.)

⁴If clicking doesn't start a file download, or you can't get the file to save into `Desktop` ▶ `lab02`, try right-click or, on a Mac, `ctrl`-click to get some sort of “Download Linked File As” or “Save target as...” or “Save Link As” option.

Now change the working directory to the folder `lab02` with this command:

```
cd lab02
```

You should see the words “Desktop” and “lab02” in the command shell prompt.

Now you should be able to run Python on one of the scripts you downloaded in Section 3.1, via

```
python verify_shell_location.py
```

(To type each of the two *underscore* characters “_” in the name `verify_shell_location.py`, on most keyboards, do `shift`+`-`, where the latter is the key you hit to get a minus-sign.)

What output do you get? **If you get anything that contains the word “WARNING” or didn’t understand something in this section, ask for help immediately before going any further in this lab.**

Later, you can refer to our webpage <http://www.cs.cornell.edu/courses/cs1110/2018sp/materials/command.php> for more on working with the command shell.

4. USING FUNCTIONS FROM A MODULE IN INTERACTIVE MODE

The previous section set up the command shell so that its working directory contains the lab files. Let’s now experiment with the functions in file `greetings.py` in `Desktop ▶ lab02`.

We’ll first use Python interactive mode, so, start it up.⁵

Make the name of the module `greetings` accessible to Python by entering `import greetings` at the “>>>” prompt. (You won’t get any output.)

See human-readable documentation of what is in `greetings` by entering `help(greetings)`, and pay special attention to the description for `multi_hi`.⁶

Do you still remember the `multi_hi` description? Probably not. So instead, since you’re starting to become a programmer:

- (1) Start up Komodo Edit.
- (2) In Komodo Edit, open⁷ file `Desktop ▶ lab02 ▶ greetings.py`, where the `Desktop` icon means your home directory.
- (3) In `greetings.py`, look for the line “`def multi_hi(name, num):`”. Underneath it is the *docstring*, enclosed in triple double-quotes, that explains what the function `multi_hi` does; when you were in Python interactive mode, when you typed “`help(greetings)`”, Python extracted that docstring and displayed it to you.⁸

⁵Reminder: enter `python` at the command shell prompt; you should then get the “>>>” Python prompt.

⁶If you find you can’t seem to escape the help documentation, hit `Q` to quit the help output.

⁷See the course webpage labeled “Python/Komodo” for instructions on how to open files: <http://www.cs.cornell.edu/courses/cs1110/2018sp/materials/python.php#komodo-open-save> .

⁸Which is kind of cool, but you might not appreciate that until later. It’s OK if you don’t find that particularly interesting now.

- (4) Also, below the docstring for `multi_hi` is the actual code body for the function. Notice that it uses the string repetition `*` we learned about in Section 2.1.

The docstring and code may give you enough information to guess what each expression below evaluates to. So, fill in your guesses below. Then, verify whether Python agrees with you!

Expression	Expected Value	If Python disagrees, why?
<code>greetings.multi_hi("Wei", 2)</code>		
<code>greetings.multi_hi("Jinx", 3)</code>		
<code>greetings.multi_hi("Potenuse", 1)</code>		
<code>greetings.multi_hi(Wei, 2)</code>		

(Where did those names come from? Think about how you would pronounce “highway”, “hijinks”, and “hypotenuse” ... Just trying to bring a little levity to this lab!)

The last expression should give you an error, because you haven’t created a variable `Wei` yet.

5. USING FUNCTIONS FROM A MODULE IN A SCRIPT

Now, let’s see how function `multi_hi` can be used in a Python script that is run by the command shell.

First, **exit Python** but **don’t** quit the command shell.

You’ll be running the Python script `run_multi_hi.py`, so open it up in Komodo Edit to see what it’s supposed to do.



Hey, look, except for using “`print`” to display the result of evaluating expressions, this script does just what you just did in Section 4. Prove it to yourself as follows: at your command-shell prompt⁹ enter `python run_multi_hi.py` in order to have Python run the script.

Notice that you get an error message stating that there’s a problem with a particular line. Read the error message to answer the following questions: Which file has the problem? Which line number has the problem? And what does that line say? Is it similar to what you saw in Python interactive mode?

⁹Which should *not* be “`>>>`”.

6. WRITING YOUR OWN FUNCTIONS

6.1. Complete function `rand_hi()`. Let’s now make `rand_hi()` in `greetings.py` do something interesting: generate a random number of “hi”s, using the function `randint` from Section 2.2.

A skeleton of function `rand_hi` is in `greetings.py`. Additionally, we’ve written a test script named `test_rand_hi.py` that will help you check your function as you go. It’s useful to be able to see both files in Komodo Edit at the same time, so here’s how you can do it. Go to (or re-open) `greetings.py` in Komodo Edit; then activate “*Split View*”: in the Komodo Edit menu,  . You’ll get two file-viewing panes. Click on the pane you want to view `test_rand_hi.py` in, and then open `test_rand_hi.py`.¹⁰

You can see that `test_rand_hi.py` just prints the result of evaluating the function `rand_hi` several times. Now, scroll down in `greetings.py` so you can see the current definition of `rand_hi` (it’s also on the last page of this handout). Right now, the function just sets variable `reps` to 0, and doesn’t have a return statement. This means that it implicitly returns the special value `None`. What do you therefore think will happen if you run Python on `test_rand_hi.py`? Test your theory out (enter `python test_rand_hi.py` in the command shell). What do you get, and why?



Let’s get to work on `rand_hi`!

The first step is to take responsibility/credit for the changes you’ll be making¹¹, as follows: go back to `greetings.py`, and in line 2, replace the text “PUT YOUR NAME AND NETID HERE” with, well, your name and net ID. Make sure you leave the comment character ‘#’ at the beginning of the line.





Now, *save and test*: save¹² the file (*This is important! Forgetting to save leads to confusion!*) and check that you didn’t insert any syntax errors by running Python on `test_rand_hi.py` again; you should get the same boring result you got before, but no errors. *If you do get an error or get different output than before, ask for help.*

Since `randint()` is in module `random`, make that module’s name available in `greetings.py` by inserting the line

```
import random
```

underneath the module docstring, i.e., around line 7 or 8. Make sure that you did *not* indent your new line. Insert blank lines so that the import statement is visually separated from both the module docstring above it and the beginning of the function definitions below it.¹³

Save-and-test again; you should get the same output as before.

¹⁰If you’d like the two files to be side-by-side instead of one-above-the-other, use the Komodo Edit menu option  . You can toggle back and forth between horizontal and vertical layout by repeated application of  .

¹¹You will need to do this for every assignment, so you might as well get into the habit now.

¹²See the course “Python/Komodo” webpage for instructions on how to save files and *check whether files have been saved* in Komodo Edit: <http://www.cs.cornell.edu/courses/cs1110/2018sp/materials/python.php#komodo-open-save>.

¹³You can look at line 7 of file `run_multi_hi.py` to see an example of an import statement in a Python file.

Now, we want `reps` to store a random number, not just be 0 all the time. So, replace the placeholder line that starts with `reps = 0` with the following:

```
reps = random.randint(1,6)
```

Make sure that your new line is still indented the way the previous version was.

Save-and-test again; you should still get the same output as before, but we need to make sure there aren't any syntax errors introduced.

Finally, add the line `return multi_hi(name,reps)`, indented the same amount as your `reps = ...` line.

Save-and-test: now what output do you get?

Why do you have to say `return multi_hi(name,reps)`, and not `return greetings.multi_hi(name,reps)`?

6.2. Use `rand_hi()` as a helper for `natural_hi()`.

The greetings produced by `test_rand_hi.py` are informal, but rather impersonal. Function `natural_hi` later in file `greetings.py` is meant to be an improvement.

Scroll down in `greetings.py` so you can see the code for `natural_hi` (it's also on the last page of this handout). Right now, it just uses a Python built-in called `input` to ask the user a question, and then stores the user's answer in variable `input_name`.

Let's change `natural_hi()` to feed the value it stores in `input_name` as an input into `rand_hi()`, and use the value that the latter returns as its own output. To do this, add the following line, indented exactly as the line `input_name = input ...` is:

```
return rand_hi(input_name)
```

Finally, run Python on `last_task.py`. If you get some personalized output plus an encouraging statement, write them here; otherwise, ask for help.

Indicate all changes you made to `greetings.py` on the copy of the code given on the last page of this handout and show it to the lab staff.

We hope you got a "natural high" from completing this lab!!

7. INDICATE CHANGES YOU MADE TO GREETINGS.PY HERE

```
1 # greetings.py
2 # PUT YOUR NAME AND NETID HERE
3 # Feb, 2018
4 # Skeleton by Prof Lillian Lee
5
6 """Library of functions producing greetings"""
7
8 def multi_hi(name,num):
9     """Returns a string of the form: "hi " repeated <num> times, followed by
10    <name>, followed by "!".
11    Preconditions (i.e., assumptions this function makes about its input):
12        name is a string
13        num is a positive int
14    """
15    return "hi "*num + name + "!"
16
17
18 def rand_hi(name):
19     """SEE LAB HANDOUT.
20    Precondition: name is a string"""
21     reps = 0 # ***placeholder: replace as instructed***
22
23
24
25 def natural_hi():
26     """SEE LAB HANDOUT"""
27     input_name = input('Please enter your name: ')
```