CS 1110, LAB 2: FUNCTIONS AND ASSIGNMENTS

http://www.cs.cornell.edu/courses/cs1110/2017fa/labs/lab2/

First Name: _____ **Last Name**: _____ **NetID**: _____

The purpose of this lab is to get you comfortable with using assignment statements, functions, and their associated modules. These are all a major part of the first assignment. The early labs are designed to get you ready for that assignment.

This lab is very similar to the previous one, in that you will be typing commands into the Python interactive shell and recording the results. As before, there are two tables. The first table challenges you to evaluate an expression. The second table asks you to "fill in the blank", completing an expression to give you the provided value.

Getting Credit for the Lab. Once again, you have a choice between getting credit through the online system or your instructor. The online lab is available at the web page

http://www.cs.cornell.edu/courses/cs1110/2017fa/labs/lab2/

The advantage of the online system is that you can do it on your own time and verify that you got credit. The disadvantage is that your answers must be correct. If you want more g uided help on this lab, you should use the worksheet instead. Despite the demands of the online system, labs are graded on effort, not correctness.

If you use this worksheet, your answer will include both a sheet of paper (or the sheet provided to you in lab) and file called `dice.py`. This is a file that you will create frome scratch. When you are finished, you should show both this file and your written answers to your lab instructor, who will record that you did it.

This lab is long enough that you may not finish during class time. If you do not finish, **you have until the beginning of lab next week to finish it**. Over the next week, you may either (1) complete the lab online, (2) show your lab to a consultant during consulting hours, or (3) show your lab to an instructor at the beginning of the next lab session. The online version will stop receiving submissions after Wednesday at midnight.

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

1. Built-In Functions

Below are two sets of tables. The first table challenges you to evaluate an expression. Use Python to evaluate the expression and type the result into the box. It is possible that the expression may produce an error. In that case, simply write `error` in the box.

The second table is looks very similar to the first table, but this time we have given you both the expression and the value. However, the expression is missing a literal, as indicated by the box. In the third column you are to fill in the missing literal. Remember that a literal is an expression that looks like a value. So `12` and `True` are literals, while (10+2) is not.

Expression	**Calculated**
`min(25, 4)`	
`max(25, 4)`	
`min(5,max(7,4))`	
`abs(25)`	
`abs(-25)`	
`round(25.6)`	
`round(-25.6)`	
`round(25.64, 0)`	
`round(25.64, 1)`	
`round(25.64, 2)`	
`len('Hello')`	
`chr(65)`	
`chr(66)`	
`ord('A')`	
`ord('AB')`	

Expression	**Calculated**	**Missing**
`min(` ☐ `,4)`	2	
`max(` ☐ `,4)`	6	
`min(4,max(` ☐ `,2))`	3	
`abs(` ☐ `)+` ☐	4	
`abs(` ☐ `)-` ☐	4	
`round(` ☐ `+0.4)`	2	
`round(` ☐ `+0.4)`	-2	
`round(5.18,` ☐ `)`	5.0	
`round(5.18,` ☐ `)`	5.2	
`round(5.18,` ☐ `)`	10.0	
`len(` ☐ `)`	2	
`len(` ☐ `)`	0	
`chr(` ☐ `)`	'C'	
`ord(` ☐ `)`	66	
`ord(chr(` ☐ `))`	121	

2. Using a Python Module

To use a module, you must import it. Type the following into the Python interactive prompt:

```
>>> import math
```

You can now access all of the functions and variables in math. However, to use any of them, you have to put `math.` before the function or variable name. For example, to access the variable pi, you must type `math.pi`.

With this in mind, fill out the tables below. The instructions for these tables is the same as for the previous activity. Read the web page for the math module if you are having trouble.

Expression	Calculated
`math.sqrt(9)`	
`math.sqrt(-9)`	
`sqrt(4)`	
`math.floor(3.7)`	
`math.ceil(3.7)`	
`math.ceil(-3.7)`	
`math.trunc(3.7)`	
`math.trunc(-3.7)`	
`math.pi`	
`math.cos(math.pi)`	
`math.acos(1.0)`	
`math.e`	
`math.log(math.e)`	
`math.log(4,2)`	

Expression	Calculated	Missing
`math.sqrt(` ☐ `)`	4.0	
`math.sqrt(` ☐ `)**2`	4.0	
☐ `.sqrt(16)`	4.0	
`math.floor(` ☐ `+0.5)`	-2	
`math.floor(` ☐ `+0.5)`	2	
`math.ceil(` ☐ `+0.5)`	2	
`math.trunc(` ☐ `+0.5)`	2	
`math.trunc(` ☐ `+0.5)`	-2	
`math.cos(` ☐ `)`	1.0	
`math.sin(` ☐ `)`	0.0	
`math.asin(` ☐ `)`	0.0	
`math.log(` ☐ `)`	0.0	
`math.log(math.e**` ☐ `)`	4.0	
`math.log(` ☐ `,2)`	3.0	

3. Variables and Assignment Statements

As we saw in class, assignment statements are different from expressions. A statement like

```
>>> b = 3 < 5
```

is a command to do something. In particular, this command

(1) Evaluates the expression on the right-hand side of the = (in this case, `3 < 5`).
(2) Stores its value in the variable on the left-hand side of the = (in this case, `b`).

Because it is not an expression, Python will not actually output a result when you type it in; it will just perform the command silently. It is important that you understand the difference.

In the first table below, the first column contains either an expression or a command. If it is an expression, write the value. If it is a command, you should just write `None` (capitalization is important here). However, you need to remember this command as it will affect the answers for later rows in the table. This is why it is very important that you *enter the expressions or commands in exactly the order they are given.*

The second table is a bit different from the previous activities. This time the first column only has assignment statements. However, these statements are either missing a literal or a variable, as indicated by the box. The second column has an expression that should be True when evaluated. You are to fill in the blank with either a literal **or a variable** that makes this expression true (this implies that all variables exist, so there are no errors). For example the answer to the first row should be `x`. Once again, assignments are assumed to be in order; answers to one row may affect later rows.

Expression	Calculated
`i = 2`	
`i`	
`j`	
`j = 1`	
`j`	
`j = j+i`	
`j`	
`i`	
`w = 'Hello'`	
`i+w`	

Expression	Calculated	Missing
`□=4`	`x == 4`	
`□=6`	`x != 4`	
`□=2`	`x == 3*y`	
`x=y+□`	`x == 8`	
`z = (□ == x)`	`z`	

4. Die Roller Script

For the last part of this lab, you will create a module/script. This is a single file that will work as both a module and a script, depending upon how you use it. This is an important exercise. The lessons you learn here will form a core part of the lab next week.

A lot of board games require you to roll two (six-sided) dice and add the results together. In this lab you will create a script that essentially does this. Of course, Python does not have any physical dice to roll. But it can generate random numbers, which is the same thing.

To understand how random numbers work in python, you should read the documentation for the module `random`.

http://docs.python.org/3/library/random.html

In particular, you want to pay attention to the function randint(a,b).

We want you create a file called `dice.py`. In that file, you should do three things:

(1) Generate two random numbers in the range 1 to 6 (inclusive)

(2) Add the numbers together and assign them to the variable `roll`

(3) Print the variable `roll` at the end.

In addition to these steps, we would like you to place a document comment at the start of the file. A document comment starts and ends with three quotes (`"""`). It also contains the following four lines:

(1) The first line is a one-line description of the module.

(2) The second line is blank.

(3) The third line your name and net-id

(4) The last line is today's date.

As we shall see in class, sometimes there are more lines between line two and three. But four is enough for now, and is typically the format we will use for this class. As a general rule, all but the last two lines will be provided for you, and you will need to write these last two lines (name, net-id, and date) yourself.

For the file `dice.py` we want the first line to be

```
A simple die roller
```

Complete the file `dice.py` before going on to the next questions.

5. Questions

While we can test your script, we want you to test it on your own. Create a folder called `lab02` and store `dice.py` in this folder. Like you did last week, open up a command shell and navigate to this folder. Start Python in interactive mode and type

```
>>> import dice
```

What do you see?

Now type the command

```
>>> dice.roll
```

Again, what do you see? How does it compare to what you saw before?

The document comment that you wrote at the top is important, as it provides the user with information on how to use the script. Type the command.

```
>>> help(dice)
```

What do you see just below the word Name (which is in bold)?

Finally, quit Python so that you are back in the general command shell from last week's lab. Now we want you to run `dice.py` as a script. In your command shell, type

```
python dice.py
```

One last time, what do you see?