## CSC 148 Lab 2: Expression and Assignments

Due: Next Friday, September 17th. The TA will check your solutions in class.

You are encouraged to discuss with your classmates, the instructor, and the TA.

Learning goals:

1. Get hands-on experience using Python in interactive mode via the command shell.
2. Get hands-on experience with Python types, expressions, and variables.

Lab 2 submission: Please turn in this file and the .py files of the programming exercise in a folder, and zip the folder as Firstname\_lastname\_Lab2 and submit it to Canvas.

**Part 1. Expressions**

The following pages have a list of expressions. For each expression, first compute the expression by head, without Python. Write down what you think the value is in the second column of the table. If you have no idea, write “?”.

Next, use Python to compute the same expression. You may find it easier to cut-and-paste from the online version of these instructions. Write down Python’s result in the third column. **You should always fill in the second and third column of a row before moving on to the next row.** You want to learn from earlier examples before moving on to the next one.

If the two values are different, you should try to figure out why Python gave the answer that it did. Come up with a reasonable explanation and put it in the final column. You are not really being graded on complete correctness in these labs (see below), so make your best guess at what is happening; your answer will help me understand how to better aid you.

Don’t be stuck with yourself. Feel free to discuss with your classmates.

If you do not finish your lab, you have until the next lab to finish it. Please show your work before you leave. If you are not done, you can show me at the beginning of the next lab. Lab is graded with effort, not correctness.

1.1 Types and Castings

Useful Shortcuts: The up-arrow key gives you the previous expression that you typed in, and you can hit it repeatedly. If you go too far back, you can press the down-arrow key to get to a later expression. The left and right-arrow keys move the text cursor on a single line. Using all of these keys together, you can take a previously-used expression, modify it, and try it again.

|  |  |  |  |
| --- | --- | --- | --- |
| **Expression** | **Expected Value** | **Calculated Value** | **If it's different, why?** |
| type(4) | Var(num) | <class ‘int’> | I thought 4 was considered a variable and had forgotten what the output would be for checking the type of value of something. |
| You did the entire row above before proceeding, right? | | | |
| type(4.0) | <class ‘float’> | <class ‘float’> |  |
| float(4) | 4.0 | 4.0 |  |
| int(4.3) | 4 | 4 |  |
| int(5.7) | 6 | 6 |  |

1.2 int and float expressions

|  |  |  |  |
| --- | --- | --- | --- |
| **Expression** | **Expected Value** | **Calculated Value** | **If it's different, why?** |
| 2\*\*3 | 6 | 8 | It was more a problem in terms of what I thought 2 to the power of 3 was equal to. I knew two asterisks meant power. |
| 2\*\*3\*\*0 | 0 | 2 | Same issue, I incorrectly guessed how this expression was computed. |
| (2\*\*3)\*\*0 | 1 | 1 |  |
| 5+2\*5 | 35 | 15 | I forgot the rule that multiplication/division happens before addition, so I added 5 and 2, which is 7 and multiplied that by 5. |
| (5+2)\*5 | 35 | 35 |  |
| -4 - -4 - -4 | 4 | 4 |  |
| 2\*\*2\*\*0 | 2 | 2 |  |
| (2\*\*2)\*\*0 | 1 | 1 |  |
| 6/4 | 1.5 | 1.5 |  |
| 6//4 | 1 | 1 |  |
| float(3//2) | 1.0 | 1.0 |  |
| 3.0//2.0 | 1.0 | 1.0 | (Note: This is, admittedly, a weird case, but for completeness’ sake we include it to point out that int division on floats converts its arguments to ints, but then converts its int answer to a float) |
| 9.0\*0.5 | 4.5 | 4.5 |  |
| 9.0\*\*0.5 | 3.0 | 3.0 |  |
| 6 % 2 | 0 | 0 |  |
| 7 % 2 | 1 | 1 |  |
| 6.2 % 4 | 2.2 | 2.2 |  |

1.3 Comparisons and boolean Expressions.

|  |  |  |  |
| --- | --- | --- | --- |
| **Expression** | **Expected Value** | **Calculated Value** | **If it's different, why?** |
| 3<5 | True | True |  |
| 3<5 and 5<3 | False | False |  |
| True | True | True |  |
| true | True | NameError: name ‘true’ is not defined | I did not suspect a difference in the casing of the letter would yield an error, but apparently case-sensitivity is important. |
| True and False | False | False |  |
| True and True | True | True |  |
| True or False | True | True |  |
| False or False | False | False |  |
| not True | False | False |  |
| not not False | False | False |  |
| not False and True | True | True |  |
| not (False or True) | False | False |  |
| (5/0==1) and False | ? | ZeroDivisionError: division by zero | Apparently, it indeed went with what came first in the “and” statement, because entering the expression in the parentheses again on its own yielded the same error. It is not legal to divide by zero apparently, and 5/0 certainly does not equal 1. |
| False and (5/0==1) | False | False |  |

Why does the last expression in the table above “work” but the one above it doesn’t?

I suspect that it’s due to the fact that with an and operator, Python will look at the variable that was entered first and automatically look for the second variable to be the same. In the case of the second to last expression, it tried to calculate (5/0==1) first but returned an error because dividing by zero is an illegal operation. In the last expression, it just took the Boolean value of False and looked for the second variable to be the exact same, ignoring the actual expression. False is not included in the second variable, so Python renders the output of False.

1.4 String Expressions.

Pay close attention to spaces and to the different types of quotation marks being used. We use both ‘(single quote) and “ (double quote). But there is a difference between double quote “ and two consecutive single quotes (‘’). You can use the print function to show the calculated value.

|  |  |  |  |
| --- | --- | --- | --- |
| **Expression** | **Expected Value** | **Calculated Value** | **If it's different, why?** |
| ‘Truth’ + ‘is’ + ‘best’ | ‘Truthisbest’ | ‘Truthisbest’ |  |
| “Truth” + “is “ +”best” | ? | ‘Truthis best’ | I was not sure what would come up, but it looks like the result calculated is due to the fact that there is a space after “is” before “best” is concatenated. |
| “Truth” + (‘is’ + “best”) | ‘Truthisbest’ | ‘Truthisbest’ |  |
| ‘A double quote: “’ | ‘A double quote: “’ | ‘A double quote: “’ |  |
| “A single quote.’” | “A single quote. ‘” | “A single quote. ‘” |  |
| ‘A single quote: “ | Error of some kind | Syntax Error | Apparently there is no second quotation mark of the same kind. |
| ‘A single quote: ‘’ | Syntax Error | Syntax Error |  |
| ‘’ + ‘ok’ | ‘ok’ | ‘ok’ |  |
| ‘’+ ‘4/2’ | ‘4/2’ | ‘4/2’ |  |
| ‘’ + 4/2 | 2 | TypeError, can’t concatenate floats to strings | I forgot that floats/integers cannot be concatenated with strings and must be converted beforehand. |
| ‘’ + str(4/2) | 2.0 | 2.0 |  |

**Part 2.Variables and Assignment Statements.**

The last part of this lab involves assignment statements. You need to know the difference between expressions, which you’ve been working with so far, and assignment statements. An assignment 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), and

(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.

In the 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”. Because some of the entries are commands, it is important that you enter the expressions or commands in exactly the order they are given.

|  |  |  |  |
| --- | --- | --- | --- |
| **Statement or Expression** | **Expected Value** | **Calculated Value** | **Reason for Calculated Value** |
| i = 2 | None | None | Just telling python to assign 2 to i, wouldn’t show anything. |
| i | 2 | 2 | Python displays the value assigned to i |
| j | Error, not defined | Error, not defined | Nothing was assigned to j yet, there’s nothing for python to display |
| j = 1 | None | None | Silently assigns 1 to j. |
| j = j +i | None | None | Silently modifies the value of j by performing addition of the value of i. |
| j | 3 | 3 | Outputs the sum of j and i as requested above. |
| i | 2 | 2 | Prints value assigned to i again, which is still the same. |
| w = ‘Hello’ | None | None | Silently assigns the string “hello” to w. |
| i + w | Concatenation error, string + int | Concatenation error, string + int | Strings (“hello”) cannot be concatenated with integers/floats (2). |

**Short Programming Exercises and others.**

2.1 Open a command shell (e.g. console in Pycharm), at the Python Interactive Prompt,

Enter this:

2/3

Circle which result you get:

0 0.6666666666 something else

**If you didn’t get the middle answer (0.6666666666), get help from a staff member immediately.**

You have installed the wrong Python.

2.2 Write a simple code that asks the user for the height in inches and return the international metric such as centimeters. Then ask the user for their height in international metric and convert it to inches.

You can use input() to get user input.

Save your question as yourname\_ lab2\_p1.py

2.3 Compute the area of a circle given its radius.

You have to first import the **math** module by:

**import math**

and you need to use math.pi to compute the area of a circle given the radius.

Save your question as yourname\_ lab2\_p2.py