**COMP 1800 – Fall 2016**

**Homework 2: Python Expressions and Variables**

**(40 points)**

Number of People: Individual. Feel free to ask me for help, or visit the Computer Science Learning Center (<http://www.memphis.edu/cs/current_students/cslc.php>).

Due: Thurs., Sept. 29 by 5:30 pm

Submission: Turn in this assignment as a hard copy; no electronic submission is necessary.

Grader: TA, Swaroop Goli ([ssgoli@memphis.edu](mailto:ssgoli@memphis.edu)). Questions about grading? Please contact him first!

1. **(12 pts, 4 each)** Determine the value of each of the following Python expressions. Also briefly explain how Python gets the result it does. Most of your credit here will be from your explanations! Results alone will be given minimal credit.  
   1. 1//2+1//3+1//4+1//5
   2. 3\*12%8+1/2\*\*3
   3. (2\*"C"+"G"\*1+3\*("T"+3\*"A"))\*2
2. **(8 pts, 4 each)** For each of the following problems, write a single Python expression that will accomplish the task for you. You don’t have to write the answer, just the Python expression that you used to get it.  
   1. You’re playing a fighting game where the damage mechanics work like this. Each attack that your character performs has a certain amount of *base damage*. The damage actually inflicted, however, depends on your opponent’s *absorption* and *armor*. Absorption gives a flat reduction to the amount of damage taken. For example, if your opponent has absorption of 5 and you hit him/her with a 20 base damage attack, s/he takes only 15 damage. Armor gives a percentage reduction to the damage taken after absorption. For example, suppose your opponent has absorption of 5 and armor of 25, and you hit him/her with a 20 base damage attack. The absorption will reduce the damage to 15. The armor of 25 further reduces this by 25%, so the final amount of damage taken by the opponent is 75% of 15, or 11.25.  
        
      Suppose you have a tough opponent with 300 health, absorption of 10, and armor of 42. Calculate the number of 50 base damage attacks that this opponent can withstand. (Leave the decimals in the answer.)
   2. Display the following design on the screen using the string concatenation and repetition operators. (Hints: Remember you can include \n within a string to represent a new line, and \\ within a string to represent the backslash character itself.)  
        
      /\/\/\/\/\/\  
      \/\/\/\/\/\/  
      /\/\/\/\/\/\  
      \/\/\/\/\/\/

/\/\/\/\/\/\  
\/\/\/\/\/\/

1. **(5 pts)** This problem illustrates some shorthand notation that you can use to change the value of a variable.  
   1. *(1 pt)* In interactive mode, write the necessary Python code to create a variable named **c** and assign it a value of 11.
   2. *(1 pt)* Run the statement c += 4. What happens to the value of c?
   3. *(1 pt)* Run the statement c \*= 2. What happens to the value of c?
   4. *(2 pts)* Rewrite the statements from parts (b) and (c) to use the syntax that we covered in class.
2. **(10 pts)** Here’s a simple “magic trick” that you can do involving numbers:  
   * Ask your unsuspecting victim to think of any number.
   * Ask the person to multiply that number by 2.
   * Ask the person to add 10 to that result.
   * Ask the person to divide that result by 2.
   * Finally, ask the person to take that result, and subtract the first number that s/he picked.
   * The number that the person ends up with is always 5. It doesn’t matter what s/he picked originally. Magic!

In script mode, write a Python program to simulate performing this trick. Your program should start by storing the original number into a variable. Then, change the value of that variable according to each step of the trick. After each step, use a **print** statement to show the result of that step on the screen.  
  
Assume that the original number will always be an integer, so the result of each step should also be an integer. Don’t use floating-point numbers anywhere!  
  
Try running your program for several values of the original number, and you should see that you always end up with 5. It even works if the original number is negative! Here’s an example of what your program might look like when you run it:  
  
The original number is: 24  
Multiply by 2 and you get: 48  
Add 10 and you get: 58  
Divide by 2 and you get: 29  
Finally, subtract your original number and you get: 5

1. **(5 pts)** Suppose you have two variables named **x** and **y**. Both variables have already been assigned values (although you don’t know exactly what those values are), and you want to swap those values – that is, you want **y**’s old value to be assigned to **x**, and vice-versa. Consider the following code, which is meant to perform this swap:   
     
    **x = y  
    y = x**  
     
   Based on what we’ve covered on how variable assignment works, why would this code not perform the swap? What does it do instead? How would you modify the code so that the two variables’ values actually do get swapped?