CS 2316
Individual Homework 3 – Bad Calculator
Due: Monday January 27th, before 11:55 PM
Out of 100 points

Files to submit: HW3.py

For help:

- TA Helpdesk
- Piazza

Notes:

- **Don't forget to include the required comments and collaboration statement**
- **Do not wait until the last minute** to start this assignment in case you run into problems

---

User Interaction

You will write a few Python functions for practice with user interaction and string and list processing. In your HW3.py file, include a comment at the top with your name, section, GTID and email, and your collaboration statement. Also include each of the following functions:

1. getExpression
2. getYesNo
3. evaluate
4. solver

Function Name: getExpression

Parameters:

- None

Return Value:

- A list containing two items:
    a. List – a list containing the numbers within the expression
    b. List – a list containing the operators in the order in which they appear in the expression

Description:

This function should ask the user to input a mathematical expression. Within this expression, you should check that there is **at least two DIFFERENT mathematical operators** (i.e. +, -, *, /). Also, you should check that **no operator is used more than two times**. You may assume that the user will **NEVER** enter parentheses. A valid expression will ALWAYS contain **four operators. You may assume that if the user enters 4 operators, then they will also enter 5 numbers.**

If the conditions above are satisfied, then you should return the data from the inputted expression as outlined below. If any of the above situations are not satisfied, you should continue to ask the user for an input until the user enters an expression that satisfies the given conditions. Once the user enters a valid expression, you should return that inputted expression re-formatted into a list that has two sub-lists. The first sub-list will be of strings that contain the numbers in the expression, and the second sub-list will be made up of the strings containing the operators. You may assume that only the four basic operators (+,-,/,*) will be used, and that all numbers will be non-negative integers.

Test Cases:

1. getExpression() ⇒

   Enter an expression: 15/3+2-1+0

   [[“15”, “3”, “2”, “1”, “0”], [“/”, “+”, “-“, “+”]] is returned

2. getExpression() ⇒

   Enter an expression: 15/2/3/1
   Enter an expression: 15+1
   Enter an expression: 15+2-3/5*1-2
   Enter an expression: 15+2-1+0-1

[[“15”, “2”, “1”, “0”, “1”], [“+”, “-“, “+”, “-“]] is returned

Function Name: getYesNo

Parameters:

- String - A String to display to the user as a prompt.

Return Value:

- True, if the user entered any variation of yes (i.e. yes, YeS, yES, etc.)
- False, if the user entered any variation of no (i.e. no, NO, nO, No)

Description:

Write a function that will ask the user for an input. If the user enters any permutation of lower and upper case letters that spells out “yes”, return True. If they enter any permutation of lower and upper case letters that spells out “no”, return False. If they enter anything else, give them the error “Not a valid answer, enter yes or no.” and then repeat the question until they give a valid answer.

Your function should be able to tolerate whitespace at the beginning and end of a line; any number of tabs, spaces, or other whitespace character may be present at the beginning or end of the line.

Test Cases:

1. getYesNo(“Solve another expression?”) ⇒

   Solve another expression? I don’t know.

Not a valid answer, enter yes or no.
Solve another expression? Yes

True is returned.

2. getYesNo(“Solve another expression?”) ⇒

   Solve another expression? nO

False is returned.

Function Name: evaluate

Parameters:

- None

Return Value:

- A float

Description:

Write a function that will evaluate a user’s expression. It should call the getExpression function that you previously wrote to get the expression to evaluate from the user. You should evaluate the expression step-by-step. For example, if the user inputs “7+5*8-2/2”, you should first evaluate “7+5”, “12*8”, “96-2”, and then “94/2”. **This function does ***NOT*** follow the usual/correct order of operations — instead, it simply reads the expression from left to right.** THIS WILL GIVE INCORRECT OUTPUT when compared to a "Good" Calculator. You should print the result of each step out to the screen and once the expression is completely evaluated, print out the final answer, like so:

Step 1: 7+5 = 12
Step 2: 12*8 = 96
Step 3: 96-2 = 94
Step 4: 94/2 = 47.0
The final answer is 47.0.

You should also return the final answer as a floating point decimal.

Hints:

1. The amount of print statements needed is going to equal the total number of operators in the expression plus one.
2. Hardcoding a separate check for each of the four operators may simplify this task, but you will lose 4 points; you should use a loop so that the same code is used four times (once per each operator) for full credit. Even though you can be assured that your code will only have to deal with 4 operators in this assignment, if you code a loop in a general way your code COULD work for any number of operators.
3. Python has a built in eval function that can evaluate string expressions, but using it introduces a potential security vulnerability in your code.

a. For example, eval("3*2") will yield 6.
b. If you complete this function **without** using the eval function, **you can earn up to 5 bonus points.**

Super Extra Bonus Points!
If you write a "good" evaluate function for the calculator (that will follow the correct operator precedence rules) and provides a correct answer for all expressions, please note this in your comments to the TA. You may NOT use the eval function to complete the "good" evaluator. Assuming it works correctly, you will receive 15 bonus points!

Function Name: solver

Parameters:

- None

Return Value:

- None

Description:

The solver function will act as the main function that will call the other functions described above. Once the solver function is called, you will need to keep track of all final answers. The following will then happen in order:

1. The user will input the expression
   a. If the expression is not valid (does not satisfy our conditions), the user will be prompted to enter a valid expression.
2. Once the user enters a valid expression, you will then look into the expression to evaluate it step-by-step.
3. Once the final answer is printed out to the screen, you should ask the user whether or not they want to enter a new expression to be solved.
   a. If yes, then proceed to repeat the process of asking the user for an expression.
   b. If no, do the following:
      b.i. Print "Your answer(s) is/are: x, y, z…" where x, y, and z are replaced by the answers to the user-inputted expressions.
      b.ii. Print "Have a good day!"

Test Cases:

1. solver() ⇒

```
Enter an expression: 15/2/3/1
Enter an expression: 15+1
Enter an expression: 15+2-3/5*1-2
Enter an expression: 7+5*8-2/2
    Step 1: 7+5 = 12
    Step 2: 12*8 = 96
    Step 3: 96-2 = 94
    Step 4: 94/2 = 47.0
    The final answer is 47.0.
Solve another expression? I don't know.
```

Not a valid answer, enter yes or no.
Solve another expression? Yes
Enter an expression: 1+1-1+1-1
Step 1: 1+1 = 2
Step 2: 2-1 = 1
Step 3: 1+1 = 2
Step 4: 2-1 = 1
The final answer is 1.0.
Solve another expression? No.
Your answer(s) is/are: 47.0, 1.0.
Have a good day!

2. solver() ⇒

Enter an expression: 1+1-0+0/1
Step 1: 1+1 = 2
Step 2: 2-0 = 2
Step 3: 2+0 = 2
Step 4: 2/1 = 2
The final answer is 2.0.
Solve another expression? No
Your answer(s) is/are: 2.0.
Have a good day!

Grading:

You will earn points as follows for each function that works correctly according to the specifications below. If the function does not work as described in the descriptions above, there will be an automatic point deduction, which is determined at the discretion of your grading TA.

getExpression — 30

Returns a list containing two lists	**5**
Checks if the user entered at least two different operators	**5**
Checks that no operator was used more than twice	**5**
Checks that the user entered exactly 4 operators	**5**
Correctly separates numbers and operators	**10**

getYesNo — 15

Returns a Boolean	**5**
Prompts user for answer until valid response is given	**5**
Checks for all permutations of valid responses	**5**

evaluate — 40

Returns a float	**5**
Correctly evaluates the expression	**10**
Prints out five print statements – first 4 are evaluations and last is final answer	**5**
Each **correct** individual operation is worth 4 points	**16**
Uses a loop to evaluate expression	**4**
EC: Does not use the eval function to evaluate the expression	**BONUS 5**
EC: Follows correct operator precedence rules!	**BONUS 15!**

(Does not stack with (or add to) the not using Eval function bonus!)

solver — 15

Keeps track of all solutions	**5**
Continues to evaluate expressions until user says to stop	**5**
Final print statement is correct	**5**