Introduction to Python: Assignment 1

Handed out: Monday, February 4, 2019.

Due: 11:59 PM, Monday, February 11, 2019

In this assignment you are going to build a simple calculator.

Submission: You should save your code assignment1.py
The first line of code should be a comment with your full name, and student number. [failure to comply would cause no credit at all]

Don't forget to include comments. Grades are partially based on readability.

Collaboration: You may work with other students; however, each student must write up and hand in his or her assignment separately. Be sure to indicate with whom you have worked in a comment at the start of each file.

Write a program that does the following in order:

1. Prompt the user for an operation with the string 'Enter operation: '
2. If the operation is exactly 'q' (without quotes), exit
3. Prompt the user for the first number with the string 'Enter first number: '
4. Prompt the user for the second number with the string 'Enter second number: '
5. Perform the operation on the number(s) and print the result
6. Repeat from step 1

Valid operations the user can type are given in the left column of the following table:

+	Add
-	Subtract
*	Multiply
/	Divide
^	exponentiate

All operations are in floating point numbers.

Sample run

```
Enter operation: +
Enter first number: 1.5
Enter second number: 2.5
4.0
Enter operation: /
Enter first number: 1
Enter second number: 2
0.5
Enter operation: -
Enter first number: 5.1
Enter second number: 0.1
5.0
Enter operation: -
Enter first number:
Enter second number: 5.1
-5.1
Enter operation: ^
Enter first number: 4
Enter second number: 0.5
2.0
Enter operation: q
```

(You may assume all inputs will be valid as described by this document.)

Extensions

The following are optional, ungraded extensions. The purpose is to help expand your understanding of Python beyond the base assignment. The difficulty of each extension is indicated by the number of ☠ icons (max of 5), relative to the material we have covered in lecture so far.

Basic Infix Expressions (☠)

The functionality of this extension is identical to the base assignment, but the input syntax is different. Instead of collecting input from three separate queries, the user should type in a single string. Shown below is the same sample as the base assignment with the input syntax updated to reflect this extension. Each operation and number will be separated by a single space.

```
Enter expression: 1.5 + 2.5
4.0
Enter expression: 1 / 2
0.5
Enter expression: 5.1 - 0.1
5.0
Enter expression: - 5.1
-5.1
Enter expression: 4 ^ 0.5
2.0
Enter expression: q
```

Longer Infix Expressions (☠☠)

Before starting this extension, you should complete the Basic Infix Expressions (☠) extension. This extension allows the user to type in more than a single operation in an expression. For the sake of simplicity, you may ignore the unary negation operation. The expressions should be computed **left to right** (i.e. there is no order of operations). Each operation and number will be separated by a single space.

```
Enter expression: 1 + 2 * 3
9
Enter expression: 1.5 + 2.5 + 1 / 2
2.5
Enter expression: 1 + 2 ^ 3 * 2 + 1
55
Enter expression: q
```

Instructions continued on the next page

PEMDAS (☠☠☠☠)

Finally, you will build a calculator that follows standard order of operations rules. In this extension, you will implement all of the operations from the base assignment as well as introduce parenthesis. The input syntax will be similar to the Longer Infix Expressions (☠☠) extension. You may consider implementing the Shunting Yard algorithm for this extension.

```
Enter expression: 1 + 2 * 3
7
Enter expression: 1.5 + 2.5 + 1 / 2
4.5
Enter expression: 1 + 2 ^ 3 * 2 + 1
18
Enter expression: 4 ^ - ( - 1 - - 1.5 )
0.5
Enter expression: q
```