ItoP: Assignment 1

Introduction

In this assignment you will build a simple calculator that performs basic operations on one or two operands. You will be introduced to console I/O, string processing, and control flow.

Directions

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 or negate (see below)
*	multiply
/	divide
^	exponentiate

All operations are on floating point numbers. Perform negation on the second number only if the user types in a blank string for the first number and the operation is '-'. Otherwise, perform subtraction.

Instructions continued on the next page

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
```

Notes

- You may assume all inputs will be valid as described by this document.
- Your output much match the sample exactly, down to the spaces and capitalization, to receive credit.

Starter files

The starter files can be downloaded from Canvas as a file called `as1.zip`. There are two files in the zip: `as1.py` and `README.txt`. You will write the program in `as1.py` where the comment directs you to and fill in the information in `README.txt`. For more information on how to run the Python program, see section Appendix.

Submission

You should submit a zip file to Canvas called `as1.zip` with the same structure as the starter files. Incorrectly structured files or an incomplete README will result in a 0.

Instructions continued on the next page

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
```

Appendix

Unlike C/C++ programs, where you must first compile the code before executing, Python files are just text run through an interpreter. The interpreter is in fact the `python` command that you run at a command line. Therefore, to run a Python program called `as1.py`, you will write the Python file in a text editor and then run it using the command `python as1.py`.

If you haven't used a text editor in the past, Sublime Text is a great starting point that works on all three major platforms. It supports syntax highlighting for most popular languages and is very extensible. If you're a little more daring, you can try using Vim. Vim is an extremely powerful text editor, but it also has a steep learning curve to the point where there are websites dedicated to teaching Vim.

Running a Python program

You can invoke the `python` interpreter using a command line. The method depends on whether you are on Windows or macOS/Linux, but the directions assume you have installed Python through Anaconda.

Windows

Search for and launch 'Anaconda Prompt'. Typing in `python --version` should print out some Python3 version. Navigate to the directory where your `as1.py` file is saved using the `cd` command and then run the program using `python as1.py`.

macOS/Linux

Open up a terminal and type in `which python`. The path that is shown should contain `anaconda3` in it. Navigate to the directory where your `as1.py` is saved using the `cd` command and then run the program using `python as1.py`.