SSEP 2024: Intro to Programming with Python

Summer 2024

Assignment 03: Clunky Calculator

*This is a **pair programming assignment find another person to work with on the assignment.***

Assignment Goals:

- Manipulate numbers with mathematical operators
- Use conditional programming structures

Notes

It is your responsibility to write a program that is well structured (i.e. modular), and easy to read. Your file should start with a header that has your name(s), the date, and a brief description of what your program does. Variable names should be descriptive. Comments should be used appropriately to document your code.

Quick Programming Tips:

- ***Not sure what to do?*** Talk through the assignment instructions with your partner and write out in English what specific tasks your program needs to do. Then, pseudocode.
- ***Stuck on how to program your solution?*** Try out 2 different ideas then ask for help if you are still stuck!
- ***Have a seemingly invisible bug?*** Use print statements throughout your code. Before running the code, think about what you expect to print if the code is working. Then see if what you expected is what prints.

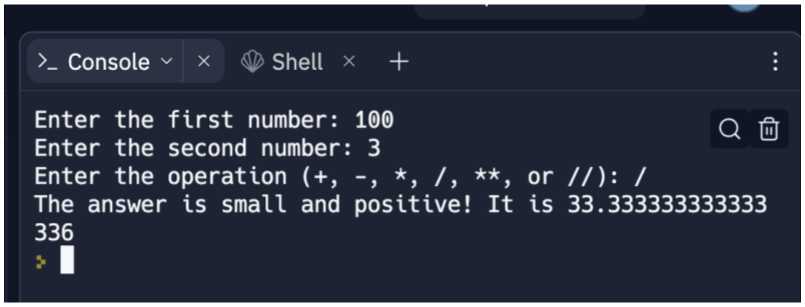
Program Specification (AKA Spec)

In this assignment, you will write a short python program that simulates basic calculator functionality.

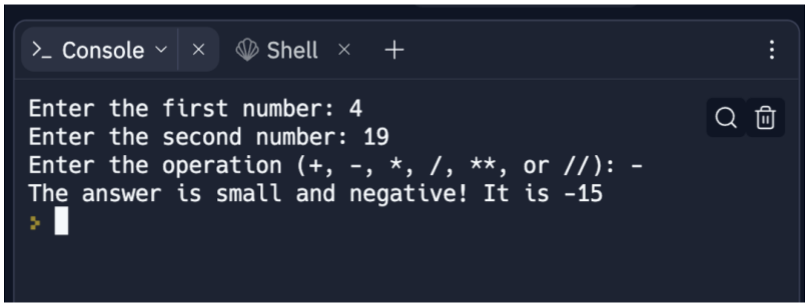
The user interface of your calculator should ask the user to input two numbers, and then a string indicating which mathematical operation to perform. It should print the answer to the requested calculation and some additional information:

- If the absolute value of the answer to the calculation is less than 100, your calculator should tell the user the answer is small. Otherwise, tell the user the answer is big.
- The calculator should also tell the user if the answer is positive or negative (or neither).

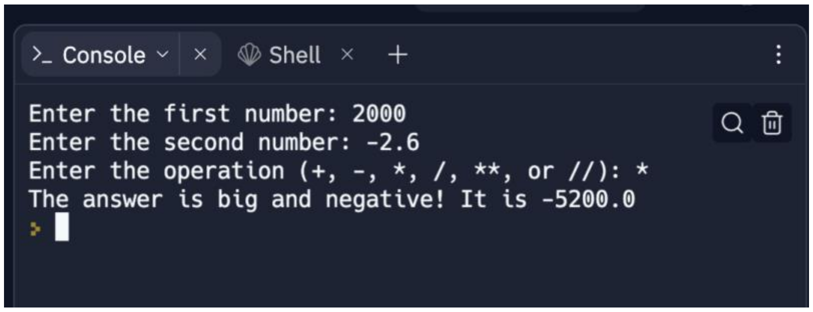
For example:



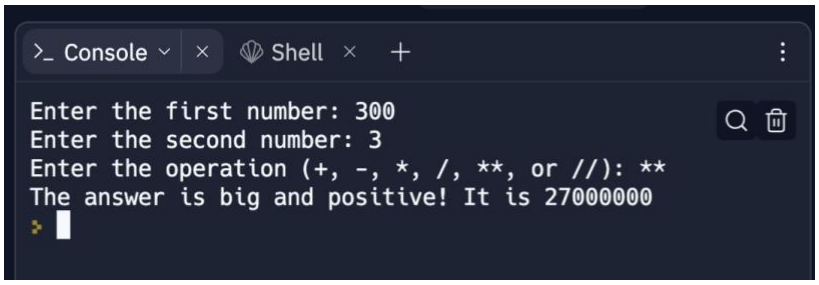
```
Enter the first number: 100
Enter the second number: 3
Enter the operation (+, -, *, /, **, or //): /
The answer is small and positive! It is 33.333333333333
336
>
```



```
Enter the first number: 4
Enter the second number: 19
Enter the operation (+, -, *, /, **, or //): -
The answer is small and negative! It is -15
>
```



```
Enter the first number: 2000
Enter the second number: -2.6
Enter the operation (+, -, *, /, **, or //): *
The answer is big and negative! It is -5200.0
>
```



```
Enter the first number: 300
Enter the second number: 3
Enter the operation (+, -, *, /, **, or //): **
The answer is big and positive! It is 27000000
>
```

Your calculator should support add, subtract, multiply, divide (both versions) and power (exponentiation) operations.

Your calculator should be able to handle both integers and floats.

Submission

Be prepared to demonstrate and discuss your code with the class. After giving every one time to work we will come together and discuss:

1. Solutions you came up with
2. Roadblocks you hit along the way
3. What you would do differently if you were to do the assignment again

Curious to know how your code would be graded in a college class? Here's an example rubric:

	Missing / Not Complete (0)	Approaching (2)	Meets (4)	Exceeds (5)
Readability	Assignment is unreadable or not submitted.	Assignment includes formatting, but significant improvements could be made. For example, headers, more documentation (comments), descriptive variable names.	Assignment includes formatting, but minor improvements could be made. For example, headers, more documentation (comments), descriptive variable names.	Assignment is well formatted and easy to read. Headers, documentation (comments), and descriptive variable names are all included.
Computational Problem Solving	No code is included in the assignment, or the code included is unreadable.	Problem solving approach could use significant improvements. Specifically, better decomposition of the problem, and/or increased modularity.	Problem solving approach is solid but minor improvements could be made with respect to decomposition of the problem, and/or increased modularity.	Problem solving approach is solid. Problem is decomposed into manageable pieces and code is modular.
Implementation	Nothing has been implemented, or most of assignment has not been done.	Code does not run consistently or efficiently. Some outputs match expected outputs. All parts of the assignment are completed except for a few small parts.	Code mostly runs consistently and efficiently. Most outputs match expected outputs, and all parts of the assignment are completed.	Code runs consistently and efficiently. Outputs match expected outputs, and all parts of the assignment are completed.