CS-500 Module Seven Activity

Isaac Raymond

CS-500-11462-M01  
  
Professor Ronak Nouri

Simple Calculator In Python - Process Summary

The goal of this project was to create a simple calculator program in Python. The program demonstrates basic functions of Python code by walking through the process of adding, subtracting, multiplying, and dividing numbers. The program demonstrates the use of user input validation, the use of conditional statements, the use of Python functions and therefore modular programming, the use of error handling in the case of division by zero, the use of loops, and the use of unit testing for verification.

The program begins by prompting the user to choose an option from a text menu. The input of the user is validated, and if found to be invalid, the user is prompted to make another entry through use of a loop in the Python code. The user could choose one of four basic calculator operations. Once the calculation has been completed, the user is prompted to decide whether to make another calculation or to end the program. This logic is mapped out using a flow chart and is attached in the project. Pseudocode is also written and provided below:  
BEGIN

DISPLAY "Select an operation:"

DISPLAY "1. Addition"

DISPLAY "2. Subtraction"

DISPLAY "3. Multiplication"

DISPLAY "4. Division"

GET operation

IF operation is not 1, 2, 3, or 4 THEN

DISPLAY "Please make another selection from the provided options"

GO TO start of program

ENDIF

DISPLAY "Enter the first number:"

GET num1

DISPLAY "Enter the second number:"

GET num2

IF operation = 1 THEN

result is num1 + num2

ELSE IF operation = 2 THEN

result is num1 - num2

ELSE IF operation = 3 THEN

result is num1 \* num2

ELSE IF operation = 4 THEN

IF num2 = 0 THEN

DISPLAY "Error: Cannot divide by zero."

GO TO start of program

ELSE

result is num1 / num2

ENDIF

ENDIF

DISPLAY "Result = ", result

DISPLAY "Would you like to perform another calculation? (Y/N)"

GET choice

IF choice = "Y" THEN

GO TO start of program

ELSE

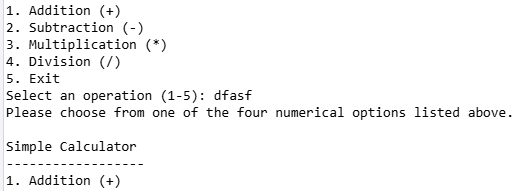
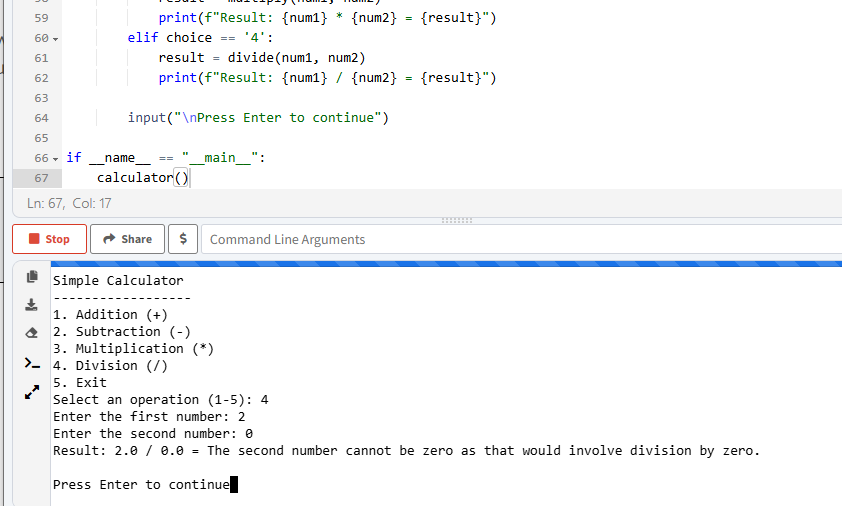
DISPLAY "Goodbye!"

END

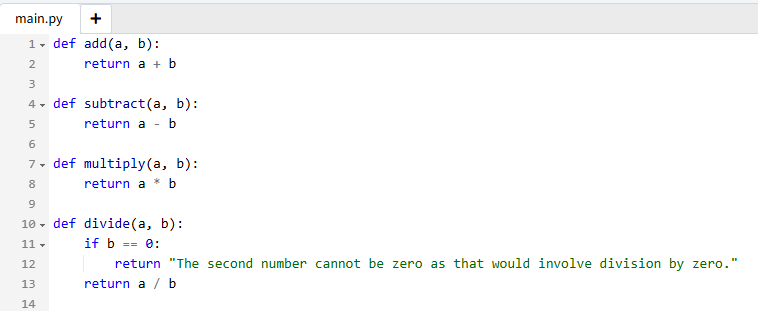
ENDIF

END

The full code of the calculator can be found in the file calculator.py. The program offers the functions of add, subtract, multiply, and divide. Each of these functions is written as a Python function in the code. When choosing one of these functions, the process of error handling is displayed as users who enter anything other than numbers are prompted to try another entry. Division by zero is also handled to prevent errors. The input is validated and any errors that are found are brought to the user's attention.

Photo: Example of input validation in the Python console:  
  
  
Photo: Example of error handling  


The use of each of these functions also provides an example of modular programming. The functions of add, subtract, divide, and multiply, can be used again if the program is expanded upon. Each of these functions are also validated using unit testing, and the unit testing file is provided. The unit testing file enters some particular values to check to make sure the expected output comes out in the end.

Photo: Example of modular programming. add(), subtract(), multiply(), and divide() can be used again if the program is expanded.  


One issue I encountered when writing the code was how to continually ask for user input when an error was made. I had to consider the possibility that a user would enter a number that was not one through 4, and then again enter another number that was not one through 4. This could not be done with simply conditional statements. Instead, I had to write a Python loop in order to accomplish the goal of allowing a user to enter their input multiple times.

another issue I encountered was and unit testing. It was my first time connecting unit testing outside of zyBooks. I wrote the unit testing Python file, but I wasn't sure how to make it interact with my original Python file. I realized that the two files need to be in the same directory and I was able to successfully conduct the unit testing.

It's difficult to imagine optimizing this program even further for performance. It's as simple as it can get. A text based calculator is very lightweight. The only way I can imagine optimizing the performance would be to somehow reduce the use of the loop, but the loop is essential to make sure that the user is allowed to continue making calculations where the user is entering the correct input. One way to optimize performance might be to stop displaying the menu after a user makes a mistake and enters a number other than one through 5. This menu is pretty important to the user experience and I would not want to exclude it.

This project demonstrates the use of Python to design and test a basic console calculator. It was an opportunity to display key programming skills such as the use of functions, the use of input validation, and the use of unit testing. Error handling and user experience were also implemented. There is obviously a lot more that could be done to expand upon the program. A graphical user interface, GUI, would make the program more practical in our modern world. More complicated functions such as exponents or square roots could be used. In the example of exponents, the modular design would be beneficial as a user could repeatedly call the multiplication function in order to calculate an exponent.