Chp.5 – Function Calculator

The goal of this activity is to create a program that has multiple helper functions that when used together, create a sort of calculator. Your calculator will support the following operations: multiply, divide, and exponents. **Even though Python has built in operators for each of these operations, the goal of this assignment is to implement them only using addition and subtraction. ANY STUDENT WHO USES A BUILT IN OPERATOR FOR MULTIPLICATION, DIVISION, OR EXPONENTS WILL RECEIVE A 0 FOR THAT FUNCTION**. Your program should be made up of the following steps:

Part 1 - Create a header comment with the title of this program, your name, and the date, along with a short summary in your own words about the purpose of your program.

Part 2 (main function)

Part 2a – Ask the user for two integer numbers. Make sure that your program stores both numbers as integers. You can assume that the user will always only give two integer numbers. You can also assume that the user will never give negative numbers. Once you run your program, the questions for the user should be formatted as follows:

```
What's the first number?
What's the second number?
```

Part 2b – Call **each** of the helper functions listed below using the user given values.

Part 2c – Lastly, call the display function to show the user the results of the various calculations completed on their given values.

Part 3 (helper functions)

Division

Part 3a – Define your division function. It should take two parameters: 1) the number being divided (i.e. the dividend), and 2) the number it's being divided by (i.e. the divisor). The function should return **two values**: 1) the number of times the first number input is evenly divisible by the second number input, and 2) the remainder value left over from the division. In the event that the second input value is zero (any number divided by 0 is undefined), the function should return "impossible" for both values. Note: To create this function, you have some limitations on what Python operators you can use. You can only use + (i.e. addition) and – (i.e. subtraction) in creating this function. You cannot use any other operators, such as * (i.e multiplication), / (i.e. division), // (i.e. floor division), etc.

Explanation

If the number being divided is greater or equal to the dividend, a division is possible. In this case, increment a counter variable by one. Then subtract the number being divided by the dividend. Assuming the number being divided is still greater than the dividend, continue the loop. Once the number being divided is less than the dividend, the remainder is assigned to the leftover value. Then return both the counter and the remainder.

Example (31 / 6)

1. 31 is the divisor, 6 is the dividend. Assume a `counter` variable = 0 and a `value` variable = 31 (the divisor)
2. (Loop iteration 1) `value` = 31 - 6 = 25; `counter` = 1
3. (Loop iteration 2) `value` = 25 - 6 = 19; `counter` = 2
4. (Loop iteration 3) `value` = 19 - 6 = 13; `counter` = 3
5. (Loop iteration 4) `value` = 13 - 6 = 7; `counter` = 4
6. (Loop iteration 5) `value` = 7 - 6 = 1; `counter` = 5
7. Since 1 - 6 will result in a negative number, the loop stops. Thus 6 fits into 31 five times with a remainder of one.

Multiplication

Part 3b – Define your multiplication function. It should take two parameters: the two numbers being multiplied together. It should have only one output: the product of the multiplication. Note: To create this function, you have the same limitations as Part 3a on what Python operators you can use. You can only use **+** (i.e. addition) and **–** (i.e. subtraction) in creating this function. You cannot use any other operators, such as * (i.e. multiplication), / (i.e. division), // (i.e. floor division), etc.

Explanation

Set the initial value of a counter to equal the first number. Utilizing a loop, use the second number as a basis for how many times the loop runs. Then the first number should simply be added to the counter value each time the loop runs. After the loop has finished, return the counter.

Example (5 * 3)

1. 5 is set as the range for the loop(The loop will run five times)
2. (Loop iteration 1) `counter(3)` += 3
3. (Loop iteration 2) `counter(6)` += 3
4. (Loop iteration 3) `counter(9)` += 3
5. (Loop iteration 4) `counter(12)` += 3
6. (Loop iteration 5) `counter(15)` += 3
7. The loop has run five times, thus it stops, and the counter is now 15

Exponent

Part 3c – Define your exponential function. It should take two parameters: 1) the number being raised to a power, and 2) the power it's being raised by. It should have one output: the result. In the event that both of the input values are zero, the output value should be set to "undefined". Note: To create this function, you have the same limitations as Part 3a on what Python operators you can use. You can however also use the helper functions you've created in Parts 3a and 3b to create this helper particular function.

Display Function

Part 3d – Define your display function. It should take four parameters: all of the values output by the helper functions you created in Parts 3a, 3b, and 3c. This function does not output anything. Its purpose is to display each of the values you calculated in Parts 3a, 3b, and 3c. When called, it should show the user these values formatted as follows (however the __(problem output #)____ will be replaced with the correct corresponding values):

```
These numbers were evenly divided a total of <3a output #1> times
This resulted in a remainder of <3a output #2>
When multiplied together, the total was <3b output>
When raised to the given power, the result was <3c output>
```

In the event that the second value given by the user is zero, division by zero is mathematically impossible, so the values displayed to the user should be formatted as follows (however the __(problem output #)____ will be replaced with the correct corresponding values):

```
Division is impossible due to a divide by zero error
When multiplied together, the total was ____
When raised to the given power, the result was ____
```

Note: All integer values should be cast as a string when being displayed.

Example Test Cases

Example Test Case:

If the user provides "10" for the first value and "3" for the second value, then your program should display the following results of the various calculations:

```
What's the first number? 10
What's the second number? 3
These numbers were evenly divided a total of 3 times
This resulted in a remainder of 1
When multiplied together, the total was 30
When raised to the given power, the result was 1000
```

Example Test Case 2:

If the user provides “10” for the first value and “0” for the second value, then your program should display the following results of the various calculations:

```
What's the first number? 10
What's the second number? 0
Division is impossible due to a divide by zero error
When multiplied together, the total was 0
When raised to the given power, the result was 1
```

Example test case number 3:

```
What's the first number? 0
What's the second number? 0
Division impossible due to a divide by zero error
When multiplied together, the total was 0
When raised to the given power, the result was undefined
```