Lab Exercise #4

Assignment Overview

This lab exercise provides practice with functions in Python.

You will work with a partner on this exercise during your lab session. Two people should work at one computer. Occasionally switch the person who is typing. Talk to each other about what you are doing and why so that both of you understand each step.

Programming with Functions

Develop a Python program that will calculate the sum of an integer series, as described below.

The program will consist of function `squares`, function `cubes`, and the main block of code.

Function `squares(initial,terms)` has two parameters: the initial integer number in the series and the number of terms in the series. It will use repetition to compute the sum of the series, and then return the results. For example, if the first parameter is 2 and the second parameter is 4, the function will return 54.

$$\text{Sum} = 2^2 + 3^2 + 4^2 + 5^2 = 54$$

Function `cubes(initial,terms)` has two parameters: the initial integer number in the series and the number of terms in the series. It will use repetition to compute the sum of the series, and then return the results. For example, if the first parameter is 3 and the second parameter is 2, the function will return 91.

$$\text{Sum} = 3^3 + 4^3 = 91$$

The program will repeatedly prompt the user to enter a command (a character string). The program will halt and display the message “Program halted normally” when the command is “exit”.

When the command is “squares”, the program will prompt the user to enter the initial integer number in the series, and then the number of terms in the series. It will call function `squares`, and then display the results.

When the command is “cubes”, the program will prompt the user to enter the initial integer number in the series, and then the number of terms in the series. It will call function `cubes`, and then display the results.

The program will display the message “*** Invalid choice ***” if the user enters an invalid command.

Function `squares` and function `cubes` will use repetition to compute the sums (rather than using a closed form equation). Hint: use `for in range()` .

You may assume that the user enters integer values when prompted for numeric values.

A suggestion: use incremental development (start with a simple version of the program, then extend it). When developing functions, it is often useful to start with a stub (a definition of the function which is syntactically complete, but logically incomplete). The following is a stub for function `squares`:

```
def squares( start, num ):
     return 0
```

The stub is a complete function definition, but it doesn't do the calculations yet.

With all that working, now make a function that can handle all exponents:
Function `power(initial,terms,exponent)` has three parameters: the initial integer number in the series and the number of terms in the series. `exponent` is the exponent for each term. It will use repetition to compute the sum of the series, and then return the results. For example, if the first parameter is 3 and the second parameter is 2, and the third parameter 3, the function will return 91. That is, if the third parameter is a 3 the function will work exactly like the `cubes` function.

★ **Demonstrate your completed program to your TA. On-line students should submit the completed program (named "lab04.py") for grading via the CSE handin system.**