Full Stack Development with AI
Lab 6.5 – Summary Exercises on Python Programming Methodology

Lab Overview

In this lab, you will combine all the Python language constructs and programming techniques that you have learnt thus far to solve intermediate to advanced programming problems.

Exercise 1 – Pascal's Triangle

Pascal's triangle is a triangular array of binomial coefficients arranged in a triangle.

To construct Pascal's triangle, we begin by numbering the first row as *n* = 0. The second row would be *n* = 1, the third row would be *n* = 2, etc. Each row contains *n* + 1 elements with the first element numbered as *k* = 0. The second element would be *k* = 1, the third element *k* = 2 until the last element, which is *k* = *n* + 1.

On row 0, we simply write the number one. To derive the elements for the subsequent rows, we use this algorithm – add the number directly above and to the left with the number directly above and to the right of a specific element to find the new number for that element. If either the number to the left or right is not present, we will substitute a zero in its place.

A Pascal's triangle of 6 rows is shown in the figure below. The first element, i.e., *k* = 0, in the sixth row, i.e., *n* = 5, is derived as 0 + 1 = 1. The second element, i.e., *k* = 1, in the sixth row is derived as 1 + 4 = 5.

```
          1
        1   1
      1   2   1
    1   3   3   1
  1   4   6   4   1
1   5  10  10   5   1
```

Mathematically, the number of each element is calculated with the same formula for calculating combinations:

$$\binom{n}{k} = \frac{n!}{k!\,(n-k)!}$$

For example, 5 *choose* 3, i.e., the sixth row (*n* = 5) fourth element (*k* = 3) in the figure, would be:

$$\binom{5}{3} = \frac{5!}{3!\,(5-3)!} = \frac{5!}{3! \times 2!} = \frac{5 \times 4 \times 3 \times 2 \times 1}{(3 \times 2 \times 1) \times (2 \times 1)} = \frac{5 \times 4}{2 \times 1} = 10$$

To simplify the computation of 5 *choose* 3, we can actually cancel out the $(3 \times 2 \times 1)$ in the denominator from the numerator. This will give us an easier formula to program in C:

$$\binom{n}{k} = \frac{\prod_{i=k+1}^{n} i}{(n-k)!}$$

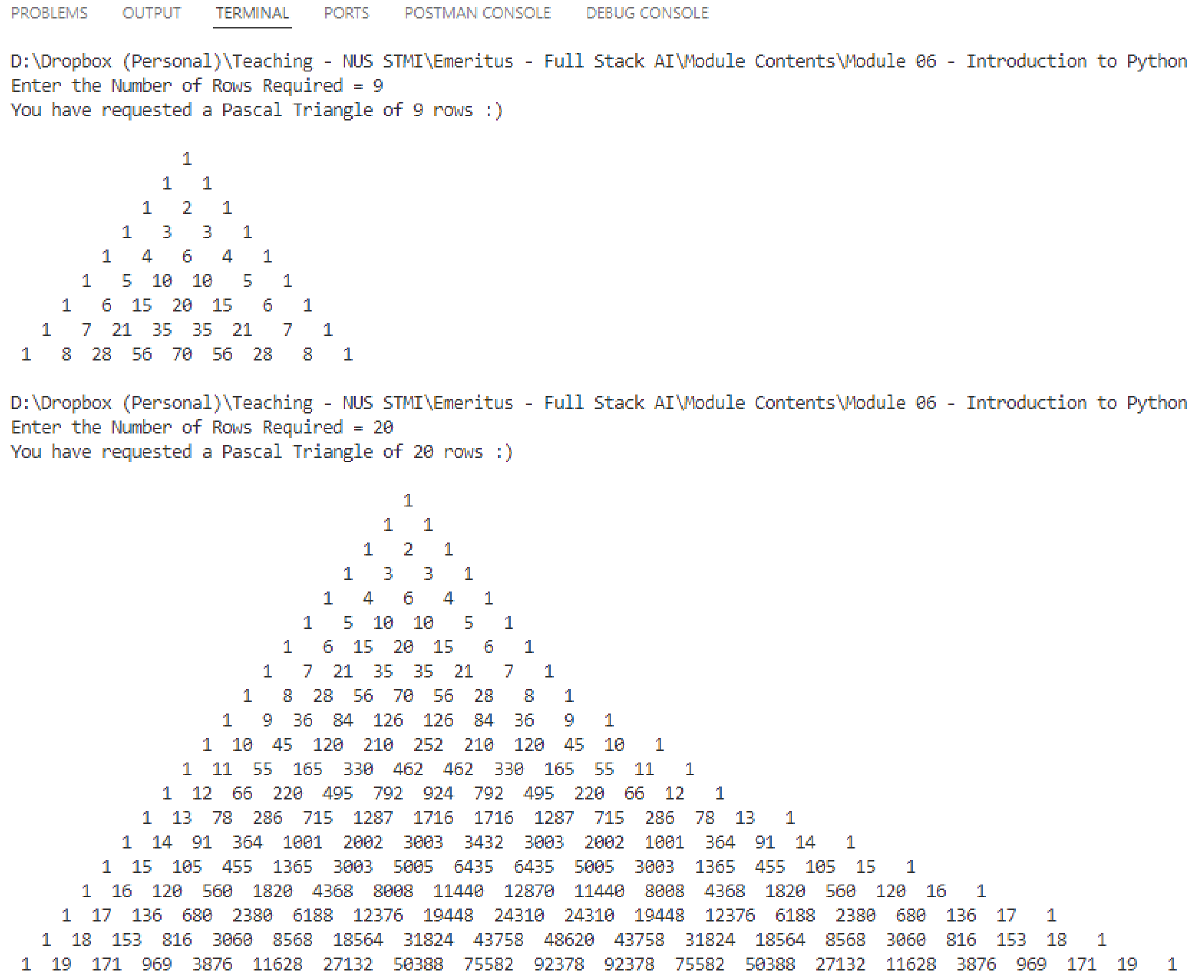
Using the simplified formula, 5 *choose* 3 would be computed as:

$$\binom{5}{3} = \frac{\prod_{i=3+1}^{5} i}{(5-3)!} = \frac{\prod_{i=4}^{5} i}{2!} = \frac{4 \times 5}{2 \times 1} = 10$$

Write a program that prompts the user for the number of rows required. Then output the Pascal's triangle with the required number of rows. The maximum number of rows that you need to handle is nine rows. If you wish to go beyond nine, don't worry if the shape of the triangle is not symmetrical.

***Hints**: The product operator for the product of a sequence, i.e., Π, can be computed using a similar algorithm as that for computing factorial.*

Sample Program Run:



```
D:\Dropbox (Personal)\Teaching - NUS STMI\Emeritus - Full Stack AI\Module Contents\Module 06 - Introduction to Python
Enter the Number of Rows Required = 9
You have requested a Pascal Triangle of 9 rows :)

                  1
                1   1
              1   2   1
            1   3   3   1
          1   4   6   4   1
        1   5  10  10   5   1
      1   6  15  20  15   6   1
    1   7  21  35  35  21   7   1
  1   8  28  56  70  56  28   8   1

D:\Dropbox (Personal)\Teaching - NUS STMI\Emeritus - Full Stack AI\Module Contents\Module 06 - Introduction to Python
Enter the Number of Rows Required = 20
You have requested a Pascal Triangle of 20 rows :)

                                        1
                                      1   1
                                    1   2   1
                                  1   3   3   1
                                1   4   6   4   1
                              1   5  10  10   5   1
                            1   6  15  20  15   6   1
                          1   7  21  35  35  21   7   1
                        1   8  28  56  70  56  28   8   1
                      1   9  36  84  126  126  84  36   9   1
                    1  10  45  120  210  252  210  120  45  10   1
                  1  11  55  165  330  462  462  330  165  55  11   1
                1  12  66  220  495  792  924  792  495  220  66  12   1
              1  13  78  286  715  1287  1716  1716  1287  715  286  78  13   1
            1  14  91  364  1001  2002  3003  3432  3003  2002  1001  364  91  14   1
          1  15  105  455  1365  3003  5005  6435  6435  5005  3003  1365  455  105  15   1
        1  16  120  560  1820  4368  8008  11440  12870  11440  8008  4368  1820  560  120  16   1
      1  17  136  680  2380  6188  12376  19448  24310  24310  19448  12376  6188  2380  680  136  17   1
    1  18  153  816  3060  8568  18564  31824  43758  48620  43758  31824  18564  8568  3060  816  153  18   1
  1  19  171  969  3876  11628  27132  50388  75582  92378  92378  75582  50388  27132  11628  3876  969  171  19   1
```

Exercise 2 – Greatest Common Divisor

The greatest common divisor (gcd) of two positive integers is the largest positive integer that divides the two numbers without a remainder. Write a program that prompts user to input two positive integers and then compute the gcd.

You are **NOT** allowed to use any function from the `math` module.

Post Exercise Thoughts: How long does your program take to compute the gcd for two numbers? If it takes longer than the blink of your eyes, try to find a faster algorithm ☺

Sample Input	Sample Output
48,18	6
1235, 255	5
55546364, 5656	404

Exercise 3 – Least Common Multiple

The least common multiple (lcm) of two positive integers is the smallest positive integer that is divisible by both numbers. Write a program that prompts user to input two positive integers and then compute the lcm.

You are **NOT** allowed to use any function from the `math` module.

Sample Input	Sample Output
4, 2	4
234, 123	9594
8968, 5687	51001016

Exercise 4 – Palindrome Checking the Easy and Fun Way

In Python, a `str` is a sequence of characters that is structurally similar to a `list` of characters except that `str` is immutable. In other word, we can only use indexing to select an individual character in a `str` but we cannot modify an individual character.

Write a program to check whether a word or phrase of any number of alphabets is a palindrome. In the case of a phrases, you should strip away all whitespaces and special characters before performing the check

Sample Input	Sample Output
Madam	Is a palindrome
Apple	Is not a palindrome
Racecar	Is a palindrome
Rotator	Is a palindrome
no lemon, no melon	Is a palindrome
Eva, can I see bees in a cave?	Is a palindrome
The brown fox jumps over the lazy dog!	Is not a palindrome

Exercise 5 – Mean Calculator

Write a program that asks user to input a list of positive numbers (including floating point numbers), as many as required, one at a time. Thereafter, the program should calculate the three mean numbers of the dataset.

The formulas of the three mean numbers are provided below:

- Arithmetic mean: $A = \frac{x_1 + \cdots + x_n}{n}$
- Geometric mean: $G = \sqrt[n]{x_1 \cdot \ldots \cdot x_n}$
- Harmonic mean: $H = \frac{n}{\frac{1}{x_1} + \cdots + \frac{1}{x_n}}$

Sample Input	Sample Output
1, 2, 3, 4, 5, 6, 7, 8, 9, 10	5.500, 4.529, 3.414
15, 55, 9, 63, 80, 100, 45, 63, 26, 75	53.100, 42.813, 30.843
3.142, 55.5, 80, 90, 10, 65.6, 75.5, 45.5, 30, 25.5	48.074, 34.161, 17.155

Exercise 6 – Concatenate Any Number of Lists Index-wise

Write a program that asks user to input any number of list of string values separately. For each list, the string values are separated or delimited by a single space character.

Then concatenate all the list index-wise, i.e., the string values at index 0 of all list would be concatenated together. Create an additional new list to contain the concatenated results starting from the 0th index item from all list, then the 1st index item, and so on till the last element. If any of the original list has insufficient items, the remaining new values would just be the original values of those list that are long enough.

Sample Input	Sample Output
H na w Ke e m a ll r e s y	['Her', 'name', 'was', 'Kelly']
I am shorter I am longer by three words I am much longer by four words	['III', 'amamam', 'shorterlongermuch', 'bylonger', 'threeby', 'wordsfour', 'words']

Exercise 7 – Reusing the Mean Calculator

Recall that in Exercise 5, you were asked to write a program that asks user to input a `list` of positive numbers (including floating point numbers), and thereafter to calculate the arithmetic mean, geometric mean and harmonic mean of the dataset.

In this question, you would be rewriting the program using the procedural programming paradigm to allow the mean calculator to be reused easily in any other future programs that you might be writing.

Perform the following tasks in order:

1) Create a Python module and name it as `mean_calculator.py`.

2) Define a function `get_number()` in `mean_calculator.py` that prompts user to input a number. If the user input an empty string, return `None` to the caller.

   If the user input a non-empty string, the input data should be validated as a valid number, either integer or floating-point number. Otherwise, user should be repeatedly prompted to re-input until a valid number or empty string has been inputted.

   Thereafter, the function should cast the input data into the correct data type, i.e., either `int` if the number does not contain the decimal point or `float` if otherwise, and return it to the caller.

3) Define a function `get_numbers()` in `mean_calculator.py` that uses `get_number()` to obtain a `list` of numbers from user and return to the caller. The list may contain a mix of integer and floating-point numbers. The function should stop prompting for new number and return the current `list` as soon as user has entered an empty string.

4) Define a function `calculate_arithmetic_mean(nums)` in `mean_calculator.py` that takes in a `list` of numbers and returns the arithmetic mean of the dataset.

5) Define a function `calculate_geometric_mean(nums)` in `mean_calculator.py` that takes in a `list` of numbers and returns the arithmetic mean of the dataset.

6) Define a function `calculate_harmonic_mean(nums)` in `mean_calculator.py` that takes in a `list` of numbers and returns the arithmetic mean of the dataset.

For task (4) to (6), you may assume that the argument is a `list` of valid numbers. If the list is empty, the functions should return `None`.

Create a new Python source file and import the `mean_calculator.py` module. Using the functions that you have defined earlier in the `mean_calculator.py` module, write a program that repeatedly prompts user to input a dataset of numbers and for each dataset, repeatedly prompts user to choose the required mean number to compute. For each valid choice, the program should print out the required mean number. The program should quit the mean calculation when user chooses to change a new dataset. The program should exit when user has no more dataset to process.

-- End of Lab --