CSE 1321L: Programming and Problem Solving I Lab

Assignment 3

Module 2 Part 2

FALL 2025

What students will learn

- Problem Solving.
- Basic Program Structure.
- Input and Output with the user.
- Write code that includes while/for loop and nested loops logic.
- Structure program to include conditional logic.

Content

- Overview
- Assignment3A: String Pyramid
- Assignment3B: Sum of Unique Products
- Assignment3C: Letter Frequency Quiz

Overview:

For this assignment, you're going to practice making logic in your code. It will include loops and nested loops. In practical terms, this means you're going to expand on the concepts from previous assignments, but also include things like while loop, for loop statements. Again, start early, practice, and ask a lot of questions.

Final note: *<u>Do not cheat</u>*

If your temptation is to look online, don't. Come see us instead and ask questions – we are here to help. Remember, you are going to have to write codes in your future job interviews, so learn it now to secure a high-paying job later.

Assignment3A: String Pyramid

Write a Python program that takes a string as input and prints it in a **pyramid pattern**, where each row shows one more character of the string. Ignore spaces in the string.

Requirements:

- Your solution must use a FOR loop exclusively.
- The input size will determine the pyramid's height and width.
- Do not use len() to find the length.
- Only iterate directly over the string.
- As always, you must follow the output format provided in the Sample Outputs.
- Use meaningful variable names where applicable.

Hints:

- You can use "**IN**" to iterate over string.

Sample Output #1:

```
[Pyramid Pattern]
Enter a string: Hello
H
He
Hel
Hell
Hello
```

Sample Output #2:

```
[Pyramid Pattern]
Enter a string: Apple Banana
A
Ap
App
Appl
Apple
AppleB
AppleBa
AppleBan
AppleBana
AppleBanan
AppleBanana
```

Assignment3B: Sum of Unique Products

Write a Python program that:

1. Takes a number n as input.
2. Finds all **pairs of numbers** (i, j) where 1 ≤ i ≤ n and 1 ≤ j ≤ n.
3. Computes the product i * j for each pair.
4. Prints the **sum of all unique products** (without using lists or dictionaries to store products).

For this task:

- You **cannot** use lists or dictionaries to store frequencies as we have not covered yet.
- Use meaningful variable names where applicable and you may add comments to explain your logic.
- Must use **nested loops**.
- Cannot use range(); simulate counting with loops.
- Cannot store products; you must **detect duplicates on the fly** using loops and variables.

Hint:

To check if a product i * j has already been counted:

1. **Think of all previous pairs (a, b)** where either:
    - a < i (all pairs from previous rows), or
    - a == i and b < j (all pairs earlier in the same row).
2. Use **nested loops** to go through these previous pairs:
    - Outer loop: a from 1 to i
    - Inner loop: b from 1 to n (but stop at b < j if a == i)
3. For each (a, b) pair, check:

    if a * b == i * j:

    # mark as duplicate as we compare the result of product not a/b or i/j. i.e. 1*4==2*2 as shown in sample output2.

Example runs are shown below. The user input is shown in **red and bold** **(Notice the difference between time & times)**

Sample Output #1:

```
[Sum of Unique Products of Pairs]
Enter a number: 3
(1,1)=1
(1,2)=2
(1,3)=3
(2,1)=2 (duplicate, ignore)
(2,2)=4
(2,3)=6
(3,1)=3 (duplicate, ignore)
(3,2)=6 (duplicate, ignore)
(3,3)=9
Sum of unique products: 25
```

Sample Output 2:

```
[Sum of Unique Products of Pairs]
Enter a number: 5
(1,1)=1
(1,2)=2
(1,3)=3
(1,4)=4
(1,5)=5
(2,1)=2 (duplicate, ignore)
(2,2)=4 (duplicate, ignore)
(2,3)=6
(2,4)=8
(2,5)=10
(3,1)=3 (duplicate, ignore)
(3,2)=6 (duplicate, ignore)
(3,3)=9
(3,4)=12
(3,5)=15
(4,1)=4 (duplicate, ignore)
(4,2)=8 (duplicate, ignore)
(4,3)=12 (duplicate, ignore)
(4,4)=16
(4,5)=20
(5,1)=5 (duplicate, ignore)
(5,2)=10 (duplicate, ignore)
(5,3)=15 (duplicate, ignore)
(5,4)=20 (duplicate, ignore)
(5,5)=25
Sum of unique products: 136
```

Assignment3C: Letter Frequency Quiz

Write a Python program that plays an **interactive "Letter Frequency Quiz"** with the user.
The program will first prompt the user to input a **sentence** (lowercase letters only, may include spaces).

1. The program will **hide the sentence**, but will tell the user how many times a specific letter appears in the sentence.
2. The user has to **guess the frequency** of that letter.
3. After each guess, the program will tell the user if their answer is **too high**, **too low**, or **correct**.
4. The quiz continues for **all unique letters in the sentence**.
5. The program ends after the user guesses the frequency of all letters correctly.
6. You **cannot** use lists or arrays, but you can use the len() function. You should only use strings and simple loops. The program must be able to interact with the user by taking input and displaying results after each guess.

For this task:

1. **Initial Input:**
    - Ask the user for a sentence. Ignore spaces for counting.
2. **Iterate over the string:**
    - Count letters - do **not use lists, dictionaries, or sets**.
3. **String Indexing / Loops:**
    - Use loops and string indexing to check occurrences.
    - More details in next section.
4. **Interactive Quiz:**
    - For each unique letter, repeatedly ask the user for the frequency until they get it correct.
5. **Output:**
    - Display feedback after each guess.
        - "Too high" if the guess is higher than the actual frequency
        - "Too low" if the guess is lower
        - "Correct!" when the guess matches the frequency
6. **Game End:**
    - Program will end once all letters are processed.

String Indexing:

In Python, you can access individual characters of a string using indexing. Each character in a string has a unique position, starting from 0.

- For example, in the word "apple", the index of the letter "a" is 0, the letter "p" is 1, and so on.
- You can use string[i] to access the character at position i in a string.
- For example, if you want to get the first letter of the word "apple", you would use word[0], which will give "a".
- Similarly, word[1] will give "p", word[2] will give "p", and so on.

Example runs are shown below. The user input is shown in **red and bold.**

Sample Output #1:

```
[Welcome to the Letter Frequency Quiz]
Enter a sentence (lowercase letters only): hello world
Guess the frequency of letter 'h': 3
Too high!
Guess the frequency of letter 'h': 1
Correct!
Guess the frequency of letter 'e': 1
Correct!
Guess the frequency of letter 'l': 1
Too low!
Guess the frequency of letter 'l': 3
Correct!
Guess the frequency of letter 'o': 2
Correct!
Guess the frequency of letter 'w': 1
Correct!
Guess the frequency of letter 'r': 1
Correct!
Guess the frequency of letter 'd': 1
Correct!
Congratulations! You completed the quiz.
```

Sample Output 2:

```
[Welcome to the Letter Frequency Quiz]
Enter a sentence (lowercase letters only): apple
Guess the frequency of letter 'a': 5
Too high!
Guess the frequency of letter 'a': 1
Correct!
Guess the frequency of letter 'p': 1
Too low!
Guess the frequency of letter 'p': 2
Correct!
Guess the frequency of letter 'l': 1
Correct!
Guess the frequency of letter 'e': 3
Too high!
Guess the frequency of letter 'e': 1
Correct!
Congratulations! You completed the quiz.
```

Submission Instructions:

- Programs must follow the output format provided. This includes each blank line, colons (:), and other symbols.
- Programs must be working correctly.
- Programs must be written in Python.
- Programs must be submitted with the correct **.py** format.
- Programs must be saved in files with the correct file name:
  - Assignment3A.py
  - Assignment3B.py
  - Assignment3C.py
- Programs (source code files) must be uploaded to Gradescope by the due date.