CSE 1321L: Programming and Problem Solving I Lab

Assignment 5

Module 5

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 methods and lists.

Content

Overview:

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

Final note: *Do not cheat*

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.

Assignment5A: Pair Difference

Write a method called **pairDifferences()** that takes in a list of integers and returns a tuple containing the **absolute differences** between every consecutive pair of numbers in the list.

Details:

- A **consecutive pair** refers to two numbers that appear next to each other in the list. For example, in [5, 10, 3, 8], the consecutive pairs are (5,10), (10,3), and (3,8).
- The **absolute difference** between two numbers is always a positive difference.
- If the list has **fewer than two elements**, return an empty tuple and display appropriate message.

Requirements:

- Prompt the user to enter a comma-separated list of numbers.
- Convert the input into a list of integers before processing.
- Process list if it has enough numbers.
- Call pairDifference() function.
- Display appropriate results.

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

Sample Output #1:

```
Enter a list of numbers: 5,10,3,8
The absolute differences between consecutive numbers: (5, 7, 5)
```

Sample Output #2:

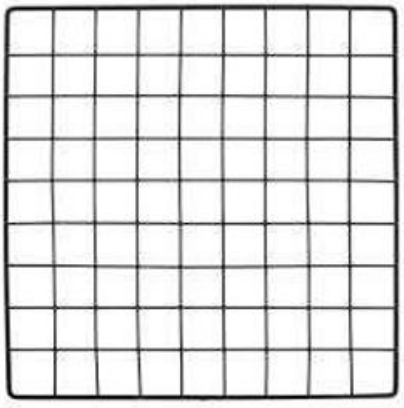
```
Enter a list of numbers: 1,-4,7,2
The absolute differences between consecutive numbers: (5, 11, 5)
```

Sample Output #3:

```
Enter a list of numbers: 42
Not enough numbers to calculate differences.
```

Assignment5B: Dungeon Treasure Map

In this assignment you will create a treasure hunt board game. Players will play on a board which resembles a grid like this:



The exact dimensions of the grid will be determined by the player. Each square of the grid will either have an 'O' representing an open square, a 'T' representing treasure, or an 'X' representing treasure that has been collected.

Steps:

- Prompt the user for the size of the grid (width and height)
- Create a list called board, each cell will represent a row of the grid.
- In the first cell of the board list, you'll place a list.
    - Pick a random number between 0 and 1. If the number is greater than or equal to 0.7 you'll add a Treasure 'T' to the next cell of the list. If the number is less than 0.7 you'll add an open 'O' to the next cell of the list.
    - Keep track of how many treasures you are adding to the board in a separate variable called numberOfUndiscoveredTreasures.
    - Repeat step (a) until you have a list that is the height the user asked for in step (1).
- Repeat step (3) until the board is the width the user asked for in step 1.
- Tell the user how many treasures you have hidden.
- Next, you'll ask the user to guess coordinates, you'll check if they found treasure or not:
    - Ask the user to enter in a row number (0 to the width of the board -1)
    - Ask the user to enter in a column number (0 to the height of the board -1)
    - Check that location to see if it is a "T" (Treasure) or an "O" (Open).
        - If it's a treasure tell the user they got treasure, change that cell of the board to an "X" to indicate that it was already discovered. Lower the number of undiscovered treasures by one.
        - If it's not a treasure, tell the user to try again.
        - Keep asking the user to guess locations until the user has discovered all the Treasures, then print out the whole board, and end the game.

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

Sample Output #1:

```
Enter the width of the grid: 2
Enter the height of the grid: 2
Number of treasures hidden: 1
Enter the row number (0 to 1): 0
Enter the column number (0 to 1): 0
No treasure here, try again!
Enter the row number (0 to 1): 1
Enter the column number (0 to 1): 1
No treasure here, try again!
Enter the row number (0 to 1): 0
Enter the column number (0 to 1): 1
No treasure here, try again!
Enter the row number (0 to 1): 1
Enter the column number (0 to 1): 0
Congratulations! You found a treasure!
O O
X O
Congratulations! You've found all the treasures!
O O
X O
```

Sample Output #2:

```
Enter the width of the grid: 3
Enter the height of the grid: 3
Number of treasures hidden: 3
Enter the row number (0 to 2): 0
Enter the column number (0 to 2): 0
No treasure here, try again!
Enter the row number (0 to 2): 0
Enter the column number (0 to 2): 1
Congratulations! You found a treasure!
O X O
O O O
O O O
Enter the row number (0 to 2): 0
Enter the column number (0 to 2): 2
No treasure here, try again!
Enter the row number (0 to 2): 1
Enter the column number (0 to 2): 1
No treasure here, try again!
Enter the row number (0 to 2): 1
Enter the column number (0 to 2): 0
No treasure here, try again!
Enter the row number (0 to 2): 1
Enter the column number (0 to 2): 2
Congratulations! You found a treasure!
O X O
O O X
O O O
Enter the row number (0 to 2): 2
```

```
Enter the column number (0 to 2): 1
No treasure here, try again!
Enter the row number (0 to 2): 2
Enter the column number (0 to 2): 0
No treasure here, try again!
Enter the row number (0 to 2): 2
Enter the column number (0 to 2): 2
Congratulations! You found a treasure!
O X O
O O X
O O X
Congratulations! You've found all the treasures!
O X O
O O X
O O X
```

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:
  - Assignment5A.py
  - Assignment5B.py
- Programs (source code files) must be uploaded to Gradescope by the due date.