Chapter 7 Lab: Lists & Tuples

Goal: This lab is intended to help you understand lists & tuples. You should hand in at least one *.py* file for each of the below problems. Please name and number the problems appropriately (something like *Lab7_1_YourName.py*). Each code should contain extensive comments in the style discussed in class and output should be clearly displayed and labeled. Note that all input/output needs to be 'protected' using *try-except* blocks.

1. Create the function *stdDev(myList)* which has

   – input: a list, *myList* , containing integer values
   – output: the mean and standard deviation (population) of *myList*

   Ask the user for a list size (no larger than 10). Then assign the list random values drawn from 1 to 9. Call the function and output the result as follows (the example is for a size 6 array):

   Your list is: [1 3 4 1 7 4]
   The mean is approximately: 3.33
   The standard deviation is approximately: 2.05

2. Create the start of the card game War.

   – Create a list of 52 cards (the deck). Each card contains a rank and a suit.
     ⋆ Rank should be a single character (we'll use "T" for 10): "2" through "9", "T", "J", "Q", "K", "A".
     ⋆ Suit should be a single character: "C", "S", "H", "D".
     ⋆ You should use nested loops to create the deck. Do not manually enter all 52 cards.
     ⋆ For example, "KH" represents the king of hearts.
   – Shuffle the deck
   – Deal 26 cards to each of the two players by giving one card to player 1, the next to player 2, then repeat.
   – You should empty the list used for the deck, transferring the cards to two separate lists (one for each player).
   – Loop through all of the player's cards. Whoever has a higher rank wins the round. If both cards have identical rank, then the round is a tie.
   – Display the number of wins for each player and the number of ties.

   You do not have to add cards to the end of the user's hand when a round is won (like in regular War).

3. Create a working game of Tic-Tac-Toe using lists (this is worth two 'normal' lab problems)

   – You board should be a two dimensional list of arbitrary dimension, $n \times n$. Allow the user to choose from $n = 3, 4$, or 5.
   – Your code must use the included function for printing. You may not alter this function, and your list must be compatible with the format found in the function (I'm trying to get you used to having to use someone else's functions).
   – Make sure the O cannot pick a spot owned by X and vice-versa.
   – Make sure the user cannot enter an invalid number or characters (for example, if $n = 3$ the only input should be integers 1 through 9).
   – You should have a function which checks to see if there is a winner. This format is up to you, but you must include all of the code that checks for a winner inside a function (not in the main code)
   – The games ends when there is a winner (print out who it is to the screen) or if there are no moves (print out that there is a tie).