Homework 0x00 - Finite State Machines and Python Basics

This assignment is to be completed individually, not in groups; however you are encouraged to work together on a solution to the assignment as long as you write your code individually.

In this assignment you will use a finite-state machine to represent a complex system and then write code to implement the logic described by your FSM. For this first assignment you will only need to run one task implemented as a finite state machine so the requirement to use only non-blocking code[1] will be relaxed for this assignment exclusively.

You may be familiar with the game called Wordle. The rules and operation of this game are similar to an older game called Mastermind played with physical pieces on a game board. Your task for this assignment will be to implement a similar game in Python to run inside a console.

In the game of Mastermind, one player, the "codemaker", starts by selecting a hidden sequence of colored pegs; the sequence must be formed by choosing 4 pegs from 6 different color choices. The second player, the "codebreaker", then tries to determine the sequence of colors by attempting to "crack the code".

In this assignment you will implement a very similar game that uses numbers instead of pegs. Your program will select a random sequence that the player must decode.

The rules of the game are as follows:

- The game board will be set up as a 4×12 grid, meaning that the player will have a maximum of 12 chances to break the code.
- The "codemaker" (your program) will start by choosing a random sequence of numbers of length 4, selected from the numeric range 0 through 5, inclusive. This means there are $6^4 = 1296$ possible sequences.
- During each turn of the game, the "codebreaker" (the program user) will enter a sequence of 4 numbers to guess the secret code. After the user enters their guess, your program will evaluate the guess and return two pieces of information: the number of correct values in the correct location, and the number of correct values in the incorrect location.

  In the traditional game, the evaluation of your guess is shown by the codemaker placing white and black pegs near the row of pegs constituting the guess. Black pegs indicate the correct color in the correct location and white pegs indicate the correct color in the wrong location, with a maximum total of 4 black and white pegs.

  In your game, use special characters to provide feedback to the user; for example, use the plus character, `'+'`, and the dash character, `'-'`, in place of black and white pegs.
- The game proceeds until the user chooses the correct sequence or runs out of rows of game board.

[1] In all multi-tasking labs it will be critical that your various tasks cooperate with each other. Blocking code - code that stops or halts operation for an extended period of time - will cause your multitasking system to misbehave.

Assignment

For this assignment you will accomplish two things:

1. Design a finite state machine to implement the rules and logic of the game. Your finite state machine design must be shown using a detailed state transition diagram utilizing the convention taught in lecture.
2. Develop a Python program, as a single .py file, that can be run from any computer to play your Mastermind-like game.

You may find it helpful to layout the game playing grid ahead of time using some "ASCII art". The following example shows what your grid might look like after one round of the game is finished.

```
Code to break: 1504

+---+---+---+---+
|   |   |   |   |
+---+---+---+---+
|   |   |   |   |
+---+---+---+---+
|   |   |   |   |
+---+---+---+---+
|   |   |   |   |
+---+---+---+---+
|   |   |   |   |
+---+---+---+---+
|   |   |   |   |
+---+---+---+---+
|   |   |   |   |
+---+---+---+---+
|   |   |   |   |
+---+---+---+---+
|   |   |   |   |
+---+---+---+---+
| 1 | 5 | 0 | 4 | ++++
+---+---+---+---+
| 1 | 5 | 0 | 3 | +++
+---+---+---+---+
| 1 | 2 | 0 | 5 | ++-
+---+---+---+---+
```

In the preceding example, the outer frame represents a Python console window; the secret code is 1504 and the player took three tries to guess the correct sequence. The grid is simply drawn using standard keyboard characters. If you are feeling ambitious, you can use the box drawing characters that are part of the extended ASCII character set to draw a very elegant game grid; see here: https://en.wikipedia.org/wiki/Box-drawing_character.

Requirements and Deliverables

This assignment will be due on October 9th. You should begin the assignment early with particular focus on the finite state machine design; however, some topics that may be useful in the assignment will be covered in lecture during the first two weeks.

In addition to implementing the basic features of the game, you must follow these additional requirements:

- The user must be able to play multiple games in a row without the program exiting. The program should keep a win-loss tally and display that to the user each time the user is prompted to start a new game.
- For evaluation purposes (and to help you with testing) the game should display the correct key sequence somewhere on screen at all times. This will greatly help in determining whether or not the logic of the game is implemented correctly.

  In order to offer an "unspoiled" version of the game, you may *optionally* choose to allow two game modes: one in which the correct sequence is displayed, and one in which it is hidden.
- The program must be completely free from errors and bugs. That is, there must be no possible behavior of your code or user input that causes your program to crash or raise an exception. All invalid user input must be handled appropriately by your program; this means that your program may need to reject or ignore certain keystrokes or provide error messages or instructions for the user if the provide invalid input.
- Your program *must* be implemented as a finite state machine, as discussed in lecture, regardless of your preferred coding style. The specific purpose of this assignment is to learn and practice with implementation of a finite state machine in software.

  If your coding structure is not formatted as a finite state machine, or alternatively does not match your state transition diagram documenting the FSM, you will receive a very large deduction to your assignment grade.

Using the software of your choice, create a state-transition diagram for the finite-state machine that you have developed. The diagram must follow the convention presented in class; that is, each state must have a number and a label shown within an ellipse, and each transition must be shown as an arrow from one state to another, labeled with a condition and optionally an action. Save your state-transition diagram as a PDF.

Once you have completed your Python program, write an extremely brief memo contextualizing your deliverables and then submit on Canvas. You will upload the the memo in PDF format along with the .py source file and your state transition diagram, also in PDF format. Be sure that you have referenced the two attachments in your memo as well.

If you are unfamiliar with memo format, please review before submitting. The memo does not need any particular formatting or department signage, it just needs to follow the appropriate conventions for memos.