**Mastermind**

Mastermind is a code-breaking game for two players: play can be simulated in text on a computer. Online versions exist and are useful for understanding how the game is played, but if you can get a hold of the actual board game, that is even better. The game is played using the following:

* A decoding board, with a shield at one end covering a row of four large holes, and 12 additional rows containing four large holes next to a set of four small holes.
* Code pegs of six different colors (we will use “colors” ABCDEF), with round heads which will be placed in the large holes on the board; and
* Scoring pegs, some black, some white, that are flat-headed and smaller than the code pegs; they will be placed in the small holes on the board. Only the quantity of black and white scoring pegs in each row matter in the game.

One player, the codemaker, selects four colors that are shielded from the other player, the codebreaker. In our version, colors cannot repeat, e.g., AABB is illegal. The goal of the game is for the codebreaker to correctly determine both the four colors selected by the codemaker and their position in the code.

The codebreaker tries to guess the pattern, in both order and color, within 12 turns. Each guess is made by placing a row of code pegs on the decoding board. Once placed, the codemaker provides feedback by placing from zero to four scoring pegs in the small holes of the row with the guess. A black scoring peg is placed for each code peg from the guess that is correct in both color and position. A white peg indicates the existence of a correct color peg placed in the wrong position.

Once feedback is provided, another guess is made; guesses and feedback continue to alternate until either the codebreaker guesses correctly or 12 incorrect guesses are made.

Write a program that simulates the game by providing the feedback. The code breaker will input each guess by entering a string of “colors”. Your simulation will ensure that guessing rules are followed: the guess consists of exactly four colors from ABCDEF. Feedback will be a count of black pegs and a count of white pegs. Your program will determine the feedback and print it. The program will declare a win if the guess is correct or a loss after 12 incorrect guesses. In addition, the program should print the complete board state so the codebreaker can more easily view the history of guesses made.

Hints:

* Play the game using paper and pencil to understand how the game is played before designing your game-playing algorithm.
* Use strings for the code and guesses.
* Use a string ‘ABCDEF’ for the set of allowable colors so you can check membership using **in**
* The isalpha string method is useful for checking input.
* The history can be built as a long string using concatenation. The end-of-line character ‘\n’ will be useful for readable output.

1. The first version of the program should prompt for the codemaker’s code. Such a game program isn’t much fun to play; but it is easier to test.
2. The final version should use the random module to create the codemaker’s code so it can be kept shielded from the codebreaker.
3. Use the index = random.randint(start, end) function to generate random indices, *start <= index <= end*, to select code characters from ‘ABCDEF’
4. Or, use random.sample(population, k) that returns a sample of length k from a specified population. The join expression (that we learn the meaning of later) convers to a string what is returned by the sample function:

code = “.join(random.sample(‘ABCDEF’, 4))