Objectives:

To practice planning, using Python sequence, decision, repetition statements, functions, files, and lists.

Description:

For this project, you will create a number guessing game. The user continues to guess until they get the number correct, then the games asks if the user wants to play again and so on.

Assignment:

1. Name your program GuessGame.py
2. Your program needs to generate an integer random number between 1 and 100 prior to guessing.
3. The user inputs a number as a guess.
4. If the user’s input value is less than the random number, output the message: “Your guess is too low.”
5. If the user’s input value is higher than the random number, output the message: “Your guess is too high.”
6. If the user’s input is equal to the random number, output the message: “Congratulations – you are correct!”
7. If the user is incorrect, steps 3 through 6 repeat until the user guesses the correct number.
8. Document your code accordingly.
9. No advanced techniques may be used, only those taught in the book.
10. You are only allowed to have two global variables – game\_history\_names and game\_history\_guesses, both lists. They store data read from the history.txt file, which keeps a history of players and their number of guesses to win the game. All other variables are to be local.
11. The history list is to be formatted. (See example run)
12. Input and output prompts should appear as shown in the example run.
13. The guess number is shown at the beginning of each guess prompt.

**Diagram

Description automatically generatedVTOC**

**Functions**

1. main() – Controls the program execution. Call main accordingly.
2. read\_history() – Receives no arguments. Opens and reads winner data from the file history.txt. The file contains the user name and number of guesses for the game they won. This data is stored in the global list variables.
3. display\_scores() – Receives no arguments. Outputs header for history list, and list data from the history file.
4. get\_player\_name() – Receives no arguments. Gets the users name, which will be stored at the end of the game in the history file. Returns name to call.
5. repeat\_game() – Receives no arguments. Asks the user if they want to continue the game and returns a Boolean value. This data is validated before returning to the calling statement.
6. store\_name() – Receives two arguments, current name and guess value and appends those to the file history.txt.

Text

Description automatically generated**Sample Run**