Programming assignment 6: Text-based Hangman

Handing in the assignment: Students hand in a single archive (**ZIP**/**RAR**/**7Z**) containing all their code (*.py*) and **storage files** on *Canvas*. Make sure all files are placed so that the program runs correctly when your main python file is run directly, and that the python file that should be run is named descriptively.
This assignment is **not** tested with automatic tests but run by humans and is mainly judged on working functionality. Points may nonetheless be deducted for:

- redundant or unnecessarily repeated code,
- messy code,
- code that is difficult to understand and not explained in comments.

Hangman
Hangman is a game of words where one player knows a word while the other player (or group of players) guesses what it is. They may guess single letters or the entire word. For each wrong guess a part of a body or gallow is added to an image and the game is lost when the entire hanged man has been drawn. The game is won if the correct word or all its letters are guessed correctly.

Your assignment is to program a text based version of Hangman. As it is difficult to draw the image your program will instead inform the player how many guesses they have left before losing the game. The computer will always play as the game master, who knows the correct word, and the user will always play as the guesser. **You can use any built-in data structure in python** and structures made by yourself, co_students or teachers in previous assignments.

The program should have access to a text file containing a ***word bank*** to choose words from. It should be possible to add words to the file and remove them with the program still running correctly. It is recommended that you read this word bank into a data structure when the program is started to have quick access to it throughout the run. Each time a new game is started a word should be randomly selected from the word bank. Then, before each guess, the program outputs the word where each letter that has not been guessed is a dash, '-', while each letter that has been guessed is in its correct place. In the beginning the word will then simply be a line of dashes. The program also outputs the number of guesses left until the game is lost.

Once a game is either won or lost the program should let the user know and offer them the possibility of either quitting or playing another game.

You can implement the following parts in any order you see fit but it is recommended to finish each group before proceeding to the next. Each part is graded on both **functionality** and **program structure**, but mainly on *functionality*.

See the next page for a breakdown of features and grading.

1. **BASIC GAME - 50%**
   - Display the word as dashes, e.g. structure = --------- ... - **10%**
   - User can guess a character and is notified whether the word contains it or not - **10%**
   - Display guessed characters, e.g. structure = -tr--t-r- after guessing 'r' and 't' - **10%**
   - Display number of guesses left each turn - **5%**
   - Detect loss when guesses are finished - **5%**
   - Detect victory when word is complete - **10%**

2. **MORE REFINED SINGLE GAME - 30%**
   - User can input or select number of guesses before the game begins - **5%**
   - After finishing a game the user can select to quit or start a new game - **5%**
   - Program stores word bank in a data structure - **10%**
   - Program randomly selects word from word bank - **5%**
   - The word bank is stored in and read from a file - **5%**

3. **CONNECTED SERIES OF GAMES - 30%** (*that makes* **110%**)
   - Keep track of wins and losses throughout the run (store in classes/variables) - **5%**
   - Allow user to guess whole word - **5%**
   - Find a way to score series of games and keep track of high scores - **5%**
     - Scores stored so that they live between runs of the program
     - Total scores can for example be affected by (*but not limited by*):
       - # of wrong guesses per word
       - # of total guesses per word (*maybe fewer if can guess whole word*)
       - # of games before loss (*or total score of those games*), etc.
   - Allow words to be added to the word bank (and file) through the program itself - **5%**
   - Allow user to see their history of games/scores - **5%**
   - Allow save/profile for multiple users - **5%**