Developers Institute

Python Course

Week 3

Day 4

Exercises

Exercise 1 – Mask Your Message

1. Create a program that will ask the user to either mask or un-mask a message. Display a menu to allow the user to choose which operation (mask or unmask) he wants to do.
2. Then allow the user to type in a message. This can be a single word or many words.
3. Masking:
    1. Once the message has been received, it will be 'masked' by interspersing it with random words from the words list file that we have used for previous exercises. For example:
        1. Message entered: "My hovercraft is full of eels."
        2. Masked message: "My green hovercraft cups is lying full exercising of cowing eels stained."
    2. Display the masked message to the user.
4. Unmasking:
    1. This is the reverse of masking. Remove every second word from the given sentence, and display the 'unmasked' message to the user. For example:
        1. Message entered: "My green hovercraft cups is lying full exercising of cowing eels antelope."
        2. Masked message: "My hovercraft is full of eels."

BONUS: Exercise 2 – Hangman

Description: We will create the 'Hangman' word-game. Here, the user will be asked to correctly guess all the letters of a randomly selected word. If she guesses too many incorrect letters, the man is hanged :(But if she guesses correctly without too many incorrect letters – she wins!

Program Flow:

1. Display a menu to the user, allowing him to play a game or exit.
2. If the user chooses to play a game, do the following steps:
    1. Select a random word (from the word list file). It must be at least 6 letters long.
    2. Give the user 10 chances to guess letters. After 10 different letters have been guessed, and if the word has not been guessed by the user, display the word, along with a message telling the user that he has lost this game.
    3. After every letter guess:

1. Check if the user has already guessed this letter. If he has, display an appropriate message, and do not decrease the number of guesses that he still has remaining. (Eg. 10 guesses – user chose ‘e’ - 9 guesses – user chose ‘e’ - 9 guesses, etc.)
2. Validate the input – it should be a single English letter only. If it’s not, show an error message, and do not decrease the number of guesses still remaining.
3. If the user has not guessed this letter before, check if the word contains this letter.
4. Display the word as a series of underscore characters (eg. “_ _ _ _ _ _ _ _ _ “), but wherever there is a letter that the user has guessed, show that letter instead of an underscore, eg. “e _ _ f _ _ e e _ r”.
5. If the user has now guessed the word completely, show a friendly message telling him that he has won this game, and how many letters he guessed.
6. If the user has not yet guessed the word completely, tell the user how many guesses he has left, and wait for the user to input a new letter.

Program Structure

- Break your code into functions and classes.
- Create a Hangman class, which will take care of:
    - reading the words from a file when the new Hangman class object is created
    - remembering all the letters that have been guessed – correct and incorrect
    - knowing if the ‘man’ is still alive (or if the hangman has won…)
    - getting a string of the word in ‘underscore form’, eg. “_ _ _ f _ _ _ _ _ r”, “e _ _ f _ _ e e _ r”.
- Create a Game class, which will take care of:
    - Creating a new Hangman class instance (object) and keeping it in a variable
    - Displaying the menu to the user – to play a new game or exit – until the user chooses to exit.
    - Asking the user to guess a letter, and validating the input (see above)
    - Passing the guessed letter to the Hangman object, and getting the result/status of the Hangman game from the Hangman object (Did the user win? How many guesses left? Did the user try to guess a letter that she already guessed earlier? Etc.)
- Create a script called run.py, which will have a short, simple main() function.
    - It will create a new Game object and call its function to begin a new game.
    - Inside the Game class’s functions, the Game object will create a new Hangman object. See above.