Hangman Game Project Documentation

Overview

The Hangman game is a simple, interactive word-guessing game where the player attempts to guess a secret word by suggesting letters within a limited number of attempts. The project is designed to help users practice control structures, file handling, and user interaction in Python.

Project Structure

Files

- words.txt: A text file containing a list of words used for the game. Each word should be separated by a space or newline.
- hangman.py: The Python script implementing the game logic.

Program Flow

- The game loads a list of words from the words.txt file.
- A random word is selected from the list.
- The player is given a certain number of chances (8 by default) to guess the secret word, one letter at a time.
- After each guess, the program provides feedback:
- If the guessed letter is in the secret word, the game displays the current progress (showing correctly guessed letters and underscores for remaining ones).
- If the guessed letter is incorrect, the number of remaining guesses decreases.
- The game ends when the player guesses the entire word or runs out of guesses.

Key Functions

1.loadWords()

This function reads the list of words from the words.txt file and loads it into memory.

Parameters: None

Returns: A list of words (strings).

Example Usage:

Python code

```
wordlist = loadWords()
```

2.Choose Word(wordlist)

This function selects a random word from the wordlist.

Parameters: None

wordlist (list): List of words.

Returns: A randomly chosen word (string).

Example Usage:

Python Code

```
secretWord = chooseWord(wordlist)
```

3. Is Word Guessed (secret Word, letters Guessed)

This function checks if the player has successfully guessed the entire secret word.

Parameters:

Secret Word (string): The secret word to be guessed.

Letters Guessed (list): The list of letters that have been guessed so far.

Re turns: True if all the letters in secretWord have been guessed; otherwise, False.

Example Usage:

Python code

```
if is WordGuessed(secretWord, lettersGuessed):
    print("Congratulations, you won!")
```

4. getGuessedWord(secretWord, lettersGuessed)

This function displays the current status of the word being guessed. Correctly guessed letters are shown, while remaining letters are replaced by underscores.

Parameters:

secretWord (string): The word the player is trying to guess.

lettersGuessed (list): The list of guessed letters.

Returns: A string representing the current guessed state of the word.

Example Usage:

Python code

```
print(getGuessedWord(secretWord, lettersGuessed))
```

5. getAvailableLetters(lettersGuessed)

This function returns the letters that have not yet been guessed.

Parameters:

lettersGuessed (list): The list of guessed letters.

Returns: A string containing all the letters that have not been guessed yet.

Example Usage:

Python code

```
print(getAvailableLetters(lettersGuessed))
```

6. hangman(secretWord)

This function controls the main game loop. It prompts the player for guesses and manages the game state (whether the player has won, how many guesses are left, etc.).

Parameters:

secretWord (string): The word to be guessed by the player.

Returns: None (prints messages to the console and manages user input).

Example Usage:

Python code

```
hangman(secretWord)
```

Game Play Instructions

1. The player is welcomed to the game.
2. The game announces how many letters the secret word contains.
3. The player guesses letters by entering them one by one.
4. If the guessed letter is part of the word, the game updates the displayed word.
5. The player has 8 attempts to guess the word. If they guess incorrectly, they lose a life.
6. If the player guesses all letters before running out of lives, they win. Otherwise, they lose and the secret word is revealed.

Sample Output

```
Welcome to the game, Hangman!

I am thinking of a word that is 5 letters long.

-------------

You have 8 guesses left.

Available letters: abcdefghijklmnopqrstuvwxyz

Please guess a letter: a

Good guess: _ _ a_ _

You have 8 guesses left.

Available letters: bcdefghijklmnopqrstuvwxyz

Please guess a letter: b

Oops! That letter is not in my word: _ _ a_ _

Congratulations, you won!
```

Possible Enhancements

- Difficulty Levels: Add options for the player to choose difficulty levels with varying word lengths or numbers of allowed incorrect guesses.
- Word Hints: Offer optional hints for the player after certain incorrect guesses.
- GUI Version: Develop a graphical user interface (GUI) for the game using libraries like Tkinter or Pygame.

Error Handling

The game checks for invalid inputs (e.g., non-alphabetic characters or already guessed letters) and prompts the player to enter a valid letter.

The program ensures the text file (words.txt) is present and accessible. If the file is missing, an error message is displayed.

Conclusion

This Hangman game project is a fun way to practice Python programming concepts like control structures, file handling, user input validation, and the use of built-in modules like random and string. It provides a solid foundation for further enhancements and learning.