PROJECT REPORT

150210310

Design, Development, and Implementation of a Hangman Game in Python

Name of the Student: Ahmet Enes Topçu

Student ID: 150210310

Summary

This report shows details the process of creating an interactive, Hangman game using the Python programming language.

The Hangman game developed for this project exhibits the basic features of a traditional game. It randomly selects a word from the word list and presents the user with an initial state consisting of empty spaces representing the letters in the word. The user progresses through the game by guessing the letters one by one. The game updates the screen after each guess, filling in correct guesses and increasing the number of incorrect attempts. The game continues as long as the user guesses the word correctly, but if they reach the maximum number of incorrect guesses, the game ends, and their score is saved.

The project uses Python's standard libraries and data structures to provide the functionality of the game. The 'Random' library plays a crucial role in making the game dynamic and replayable by choosing a different word from the list for each new game. Basic Python data types, especially strings, and lists, are used extensively for data management and game state tracking.

The game demonstrates basic Python programming concepts such as loops, conditionals, and functions. It also showcases good coding practices such as modularity through the use of functions, effective data management, and code readability for maintainability.

The report concludes by discussing the current status of the project, describing its successful implementation, and suggesting potential improvements for future versions. These improvements include expanding the word list for more variety, introducing a multi-player system, and improving the hint system.

TABLE OF CONTENTS

Hangman Game Report

Introduction

"Hanguess" is a python based interactive word guessing game inspired by the classic hangman game. It is using the Tkinter library in Python, which provides a graphical user interface (GUI) for the user. The game allows players to choose random words from various categories (such as animals, films, car brands and more) and try to guess the hidden word by entering letters or the whole word. This report aims to analyze the features and components of the game, discussing its functionality, implementation details, user interaction, evaluation, and recommendations for future enhancements.

Features

The Hangman Game offers several notable features that enhance the gameplay experience:

Word Categories

The game includes various categories of words to give players a variety of options. These categories include animals, countries, soccer teams, and more. Players can guess the type of word they want by selecting their preferred category before starting the game.

Hint System

To help players with the word guessing part, the Hangman Game includes a hint system. Players can use hints to reveal a correct letter in the hidden word and increase their chances of guessing correctly. However, the number of hints allowed is determined by the word length and using hints will reduce your points

Scoring and Leaderboard

The game has a system that calculates players' points based on their performance in each round. Players earn points for each correct guess and can lose points for incorrect guesses or using hints. The game also displays a leaderboard of the top 10 performers, encouraging friendly competition among players and motivating them to improve their skills.

Customizable GUI

Hangman Game offers a customizable graphical user interface (GUI) to enhance the visual appeal and user experience. Players can personalize the GUI by choosing different colors for various components such as the word to guess, logo, hangman figure, etc. This customization feature allows players to create a personalized gaming environment according to their preferences.

Methods

In this section, we detail the approach, techniques, and Python features utilized to implement the Hangman game.

Initialization

At the start of the game, necessary initializations are performed. A list of words is defined from which one word will be randomly chosen for each round of the game. Python's random library is used for this purpose. The chosen word is stored as a string and a separate 'hidden' version of this string is created where all characters are initially represented as underscores. This hidden string is used to track the player's progress and is updated as they guess letters correctly.

Game Loop

The main game loop is implemented using a while loop. Inside this loop, the current game state is printed, a player's guess is taken, and the game state is updated based on this guess.

Taking Player Input

Player input is taken using the built-in input() function. The input is sanitized by converting it to lower case using the lower() string method. Error handling is implemented to ensure that the input is a single letter and it has not been guessed before.

Game State Update

After each guess, the game state is updated. If the guessed letter is in the word, all occurrences of the letter are revealed in the 'hidden' string. This involves using a loop to iterate over each character in the target word and checking if it matches the guessed letter. The enumerate() function is used here to get both the index and value of each character.

If the guessed letter is not in the word, a counter keeping track of incorrect guesses is incremented. Python's built-in string and list data structures play a critical role in managing and updating the game state.

End of Game

The game continues until either the word is completely guessed (there are no more underscores in the 'hidden' string) or the player has made a certain number of incorrect guesses. Depending on the outcome, an appropriate message is displayed to the player.

The entire game is encapsulated in a function so that it can be replayed easily. This demonstrates the use of functions for code modularity, one of the key good practices in programming.

This method of implementation exhibits use of Python's fundamental programming constructs including loops (for, while), conditionals (if, elif, else), string manipulation methods (lower(), replace()), list operations, and error handling (try, except).

Discussion

In this section, we analyze and discuss the structure of the Hangman game app and the thought process behind it.

Hangman is essentially a guessing game involving a word that must be guessed letter by letter. This game is a problem of recursively updating the state of a system (in this case, the guessed word) based on inputs (the player's guesses) and uses python to do so.

A central aspect of the application is the game loop. It uses a while loop to continuously request input from the user and update the game state until a termination condition is met. This is a system that is widely used not only in games but also in the development of interactive applications.

Using a hidden version of the word that is revealed step-by-step is a practical approach to track the user's progress and update the game state. Python's flexibility with string and list data structures significantly simplifies the process of creating and updating this hidden word.

Receiving player inputs and handling errors is another important aspect of this project. It demonstrates the practical use of Python's try-except blocks for error handling. The idea is to prevent the program from crashing when a user enters something unexpected, instead giving them feedback to correct their input.

One of the interesting features of this code is the use of Python's random module to pick a random word from a list. This gives the game replayability as the word to guess can be different each time. It also demonstrates the usefulness of Python's extensive standard library.

While the implementation of this project is simple, it demonstrates numerous Python features and good practices in software development. These include iterative development using loops, state management using data structures, user interaction handling, error handling and code modularity. The selected methods and techniques used in this project are widely applicable in many different types of software development, making this project a good practical exercise in Python programming.

Implementation Details

This section delves into the technical implementation details of the Hangman Game:

Programming Language and Framework

Hangman Game is developed using the Python programming language, taking advantage of its flexibility and ease of use for game development. The game uses the Tkinter library, a standard Python interface, to design and build the graphical user interface.

Modules and Packages

The Hangman Game includes external modules and packages to enhance its functionality and gameplay. For example, it uses modules that provide ASCII art representations of the hangman stages and show the progressive state of the hangman figure as incorrect guesses are made. In addition, the game includes packages that provide previously defined word lists for different categories and allow for a variety of word choices.

Class Structure

The game follows a structured class-based approach to organize the codebase. The main logic and functionality of the game is encapsulated within a Hangman class that manages game states, tracks player progress, processes user input and updates the GUI accordingly. The use of classes supports code reactivity, modularity and maintainability.

User Interface Design

The graphical user interface (GUI) of the Hangman Game is designed to be visually interesting. The GUI consists of various components such as a text box to display the hidden word, buttons for letter selection, a scoreboard to show the player's score, and a hangman figure to show the progress of the game. The design is prepared to provide a catchy user experience.

User Interaction

Hangman Game offers a simple and interactive user interface, allowing players to easily get started with the game:

Starting a Game

When the game is launched, players are presented with a start screen where they can select a word category and customize the GUI appearance after entering their username. Once the game settings are configured, players can start a new game session.

Gameplay Flow

During the game, players are asked to guess individual letters by clicking on the corresponding buttons or using keyboard input (they can also guess the whole word from the keyboard input section.) The game validates the input and updates the GUI accordingly, revealing correct guesses and penalizing incorrect ones. The player's score and the hangman figure are updated according to their progress.

End of Game

The game continues as long as the player guesses the word correctly but ends when they have finished the allowed number of attempts. In both cases, a message is displayed to inform the player about the result. If the player achieves a high score, his/her name is included in the leaderboard.

Recommendations

The following suggestions can be made for the development of the Hangman Game:

Expanded Word Categories

Adding different categories of words, such as TV series, sports, history or science, can increase the game's charm and provide more varied gameplay experiences. This can be achieved by adding external word lists or applying online word import mechanisms.

Multiplayer Functionality

The addition of a multiplayer mode where players can compete or cooperate with each other to guess the hidden word could improve the social side of the game. This could be achieved through network programming or by applying a turn-based game feature.

Improved Hint System

Improving the hint system by offering different types of hints, such as showing multiple letters or giving word definitions, can make the game more suitable and fun for players of all skill levels. Balancing the use of hints with appropriate penalties ensures a fair experience.

Enhanced Graphics and Sound Effects

Adding visually attractive graphics, animations and sound effects can increase the gameplay. These could include dynamic hangman figure animations, attractive background themes and sound effects for correct and incorrect guesses.

Mobile Compatibility

Improving the game's accessibility by developing versions suitable for mobile platforms such as iOS and Android can increase the number of players and enable users to access the game from their smartphones and tablets. This requires adapting the GUI and input mechanisms to touch interactions.

Conclusion

In conclusion, the Hangman project in Python demonstrates the application of various core programming concepts and Python's capabilities. Key characteristics such as user interaction, state management, and error handling have been effectively used, making it a valuable learning tool for Python programming.

Python's built-in features, such as list and string manipulation, the random module, and error handling through try-except blocks, are put to use successfully, emphasizing Python's strengths in managing interactive applications.

The project underscores the significance of error handling and the benefits of modular programming through the use of try-except blocks and function encapsulation respectively.

The iterative development approach evident in the game loop highlights a common pattern in many interactive software applications.

Overall, the Hangman project is a robust example of Python's functionalities and good software development practices, serving as a useful reference for interactive application development.