**Learn Words While You Play – A Python Hangman Game**

Project Report Submitted and Requirements for Award of The Degree of

BATCHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND TECHNOLOGY

Submitted by

**Mr. M ANIL**

([ma6039564340@gamil.com](mailto:ma6039564340@gamil.com))

DECLARATION:

I declare about the project work entitled “**Learn Words While You Play – A Python Hangman Game”** is a result of my own work carried out during the course of my academic study. This project has been developed using Python and focuses on backend logic to create an educational word-guessing game.

**OBJECTIVE:**

To develop a backend-only Hangman game using Python that enhances vocabulary through hints and structured word selection.

**ABSTRACT:**

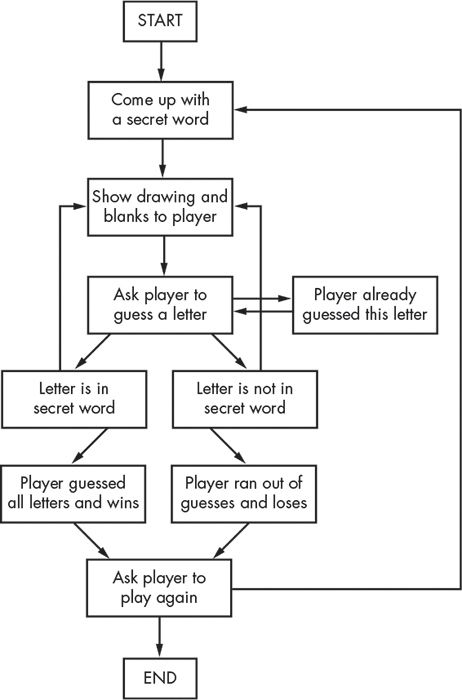
This project presents a command-line Hangman game developed using Python, designed to enhance vocabulary through interactive gameplay. Each word in the game is paired with a contextual hint to guide the player. Built entirely with backend logic, the system features modular code structure, input validation, word length display, prevention of word repetition, and a replay mechanism. The game runs in the terminal and emphasizes clean logic, educational value, and a smooth user experience. It demonstrates how backend programming can be used to create engaging and instructive tools for learning.

**INTRODUCTION:**

“Learn Words While You Play – A Python Hangman Game” is a backend-driven educational tool developed using Python. The project reimagines the classic Hangman game as a vocabulary-building experience, where each word is paired with a contextual hint to support learning. Designed to run entirely in the command-line interface, the game emphasizes clean modular code structure, and user-friendly interaction.

**DESIGN METHODOLOGY:**

1. **Word and Hint Setup**
   * Words are stored with corresponding hints in a dictionary or external file.
   * The system randomly selects a word and its hint for each round.
2. **Game Initialization**
   * Variables for missed letters, correct guesses, and used words are initialized.
   * The word length is displayed to guide the player.
3. **Input Handling**
   * The player is prompted to guess a letter.
4. **Game Logic Execution**
   * If the guessed letter is correct, it’s added to the correct guesses.
   * If incorrect, it’s added to missed letters and the hangman state updates.
5. **Win/Loss Detection**
   * The game checks if all letters are guessed (win) or if the maximum number of incorrect guesses is reached (loss).
6. **Replay Mechanism**
   * After each round, the player is asked whether they want to play again.
   * Used words are tracked to avoid repetition in the same session.

****

**TESTING PROCESS:**

**1.**  **Manual Testing**

* Run the game multiple times with different inputs.
* Try edge cases like:
  + Entering numbers or symbols
  + Repeating the same letter
  + Guessing all correct letters
  + Losing intentionally to test the hangman drawing
* Check if replay logic and word repetition prevention work properly.

**2**. **Functional Testing**

You can write small test functions to check:

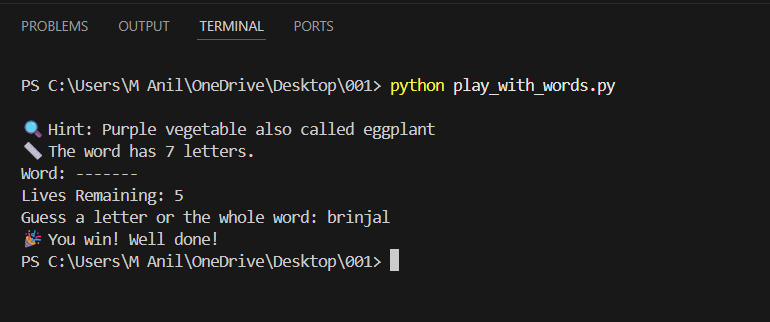
* Word selection logic
* Hint retrieval
* Input validation
* Win/loss condition triggers

**SOURCE CODE:**

**You can view the full implementation here: “play\_with\_words.py”**

This file contains the complete backend logic for the Hangman game, including word selection, hint display, input validation, replay functionality, and session management. The code is modular and well-commented to support readability and future enhancements.

**Playing the Game:**

****

**LANGUAGE REQUIREMENTS:**

This project is developed using the Python programming language, chosen for its simplicity, readability, and strong support for backend logic and educational tools.

* **Primary Language**

1. Python 3.x

* Used for all backend logic
* Supports modular code structure, input validation, and random word selection. Ideal for command-line applications and beginner-friendly syntax

**FEATURES:**

* **Hint System**

Each word comes with a contextual hint to guide the player and support

Vocabulary learning.

* **Word Length Display**

The number of letters in the word is shown at the start of each round to help

players strategize.

* **Input Validation**

Ensures only valid single-letter inputs are accepted, with clear prompts for incorrect entries.

* **No Word Repetition**

Tracks used words to prevent repeats during a session, keeping gameplay fresh.

* **Educational Focus**

Designed to reinforce vocabulary through gameplay, making learning interactive.

**FUTURE AND SCOPE:**

This project lays the foundation for a scalable, educational word game built entirely with Python. While the current version runs in the terminal and focuses on backend logic.

**Educational Enhancements:**

* **Difficulty Levels**: Introduce beginner, intermediate, and advanced word sets.
* **Multilingual Support**: Add words and hints in regional languages like Telugu or Hindi.

**Technical Upgrades:**

* **Web Deployment**: Convert the game into a web app using Flask or Django, allowing online play.
* **Database Connectivity**: Store user scores, word history, and progress using SQLite or PostgreSQL

.

**Analytics & Feedback:**

* **Performance Tracking**: Show players how many words they’ve learned or missed.
* **Hint Effectiveness**: Analyze which hints lead to faster correct guesses.

**CONCLUSION:**

The development of *Learn Words While You Play – A Python Hangman Game* demonstrates how backend programming can be used to create meaningful, interactive learning experiences. By combining classic game mechanics with vocabulary-building features like hints and word-length display, the project offers both entertainment and educational value.