****

***School of Mechanical & Manufacturing Engineering (SMME),***

***National University of Science and Technology (NUST),***

***Sector H-12, Islamabad***

Program: BE-Aerospace Section: AE-01

Session: FALL 2024 Semester: 3RD

Course Title: Fundamentals of Programming (II)

Project Report

***“Hangman Game*”**

***Name: Sharjeel Asad***

***CMS:*** 454248

**Objective:**

This project aims to create a Python-based Hangman game that enhances user engagement, reinforces programming concepts, and demonstrates efficient logic, error handling, and progress tracking through a user-friendly interface.

****Abstract:****  
This Python project implements a Hangman game, a word-guessing challenge designed to enhance logical thinking and programming skills. The game randomly selects a word from a predefined list of aerospace-related terms, and players attempt to guess it within a limited number of attempts. Key features include input validation, tracking of guessed letters, and real-time feedback on progress. The project demonstrates proficiency in Python programming concepts such as loops, conditional statements, string manipulation, and data structures like sets. This interactive game showcases the application of problem-solving and software development principles in a fun and engaging format.

**Introduction:**

The Hangman game is a classic word-guessing game implemented in Python, where the player attempts to guess a hidden word by suggesting letters within a limited number of attempts. The game features a simple text-based interface that displays the word’s progress and remaining attempts. A random word is selected from a predefined list, ensuring variety in gameplay, and user input is validated to handle repeated or invalid guesses.This project demonstrates fundamental programming concepts such as conditionals, loops, string manipulation, and randomization. The program dynamically updates game status, flags repeated guesses, and provides real-time feedback to enhance user engagement. Its modular code structure ensures readability and potential scalability for features like difficulty levels or graphical interfaces. The project combines interactive gameplay with educational value, showcasing Python’s versatility in a practical and entertaining way.

**Explanation:**

The detail explanation of code is:

### ****1. Importing Libraries****

**import random:** The random module is imported to randomly select a word from a list for the player to guess.

1. **Defining Functions:**

**get\_word()**

def get\_word(): words = ['aerospace', 'engineering', 'airplane', 'propulsion', 'satellite', 'navigation', 'gravity', 'velocity']

return random.choice(words)

* **Purpose**: To randomly select a word from a predefined list of words (words)
* **Logic**: random.choice(words) picks and returns one word at random.

**Example**:

* Word: "aerospace"
* Guessed letters: {'a', 'e', 'o'}
* Output: "ae\_o\_\_a\_e"

### ****3.Main Game Function:**** play\_hangman()

#### ****Game Initialization****

* A random word is selected using the get\_word() function.
* Several variables are initialized:

1. **guessed\_letters:** A set to track the letters correctly guessed by the player.
2. **remaining\_attempts**: The number of wrong guesses the player can make before losing (set to 6).
3. **guessed\_wrong:** A set to keep track of the letters guessed incorrectly.

#### ****4.Game Loop****

* The game continues running until the player runs out of attempts (remaining\_attempts > 0) or guesses the word correctly.
* Each turn, the following steps are performed:

1. **Display Game Status**:

* The current progress of the word is shown using display\_word().
* The number of remaining attempts and incorrect guesses are displayed

1. ****Player Inpu**t**:

* The player is prompted to guess a letter.
* Input validation is performed:

1. If the input is not a single letter or is nonalphabetical,the player is notified, and the loop continues.
2. If the letter has already been guessed (correctly or incorrectly), the player is notified.

3.**Check Guess:**

· If the guess is correct (the letter is in the word):

* The letter is added to guessed\_letters.
* The game checks if all letters in the word have been guessed (using set(word) == guessed\_letters). If so, the player wins.

· If the guess is incorrect (the letter is not in the word):

* The letter is added to guessed\_wrong.
* The number of remaining attempts decreases by 1.

4.**End of Game**:

* If the player uses all attempts (remaining\_attempts == 0), the game ends with a loss, and the correct word is revealed.

#### ****Win/Loss Scenarios:****

* ****Win**:**

1. The player successfully guesses all letters in the word before running out of attempts.
2. A congratulatory message is displayed with the correct word.

* **Loss**:

1. The player exhausts all attempts without guessing the word.
2. A message reveals the correct word.

**Code:**

import random

def get\_word():

words = ['aerospace', 'engineering', 'airplane', 'propulsion', 'satellite', 'navigation', 'gravity', 'velocity']

return random.choice(words)

def display\_word(word, guessed\_letters):

display = ''

for letter in word:

if letter in guessed\_letters:

display += letter

else:

display += '\_'

return display

def play\_hangman():

word = get\_word()

guessed\_letters = set()

remaining\_attempts = 6

guessed\_wrong = set()

print("Welcome to Hangman!")

while remaining\_attempts > 0:

print(f"\nWord: {display\_word(word, guessed\_letters)}")

print(f"Remaining attempts: {remaining\_attempts}")

print(f"Wrong guesses: {', '.join(guessed\_wrong)}")

guess = input("Enter a letter: ").lower()

if len(guess) != 1 or not guess.isalpha():

print("Invalid input, enter a single letter.")

continue

if guess in guessed\_letters or guess in guessed\_wrong:

print("You already guessed that letter.")

continue

if guess in word:

guessed\_letters.add(guess)

if set(word) == guessed\_letters:

print(f"Congratulations! You guessed the word: {word}")

break

else:

remaining\_attempts -= 1

guessed\_wrong.add(guess)

if remaining\_attempts == 0:

print(f"Game over! The word was: {word}")

break

play\_hangman()

