**AI-LAB – ASSIGNMENT#1**

**NAME: ABDULLAH**

**ROLL NO: Su92-BSAIM-F24-002**

**SECTION: BSAI-3A**

Hangman Game (OOP Based) - Step by Step Explanation

# Step 1: Basic Structure of Class

We create a Hangman class with:  
- A word to guess.  
- A list of guessed letters.  
- Number of lives (chances).  
- Hangman drawing stages (printed with each wrong guess).

Code:

class HangmanGame:  
 def \_\_init\_\_(self, word):  
 self.word = word.upper()  
 self.lives = 6

self.guessed\_letters = []  
 self.hangman\_stages = [  
 """  
 ------  
 | |  
 | O  
 | /|\\  
 | / \\  
 |  
 """,  
 """  
 ------  
 | |  
 | O  
 | /|\\  
 | /   
 |  
 """,  
 """  
 ------  
 | |  
 | O  
 | /|\\  
 |   
 |  
 """,  
 """  
 ------  
 | |  
 | O  
 | /|  
 |   
 |  
 """,  
 """  
 ------  
 | |  
 | O  
 | |  
 |   
 |  
 """,  
 """  
 ------  
 | |  
 | O  
 |   
 |   
 |  
 """,  
 """  
 ------  
 | |  
 |   
 |   
 |   
 |  
 """  
 ]

Step 2: Display Current State

We write a function to display the word dynamically:  
- If a letter is guessed, it shows.  
- Otherwise, an underscore '\_' is shown.

Code:

def display\_word(self):  
 return " ".join([letter

if letter in self.guessed\_letters else "\_"

for letter in self.word])

# Step 3: Print Hangman Drawing

We add a function to print the current hangman drawing based on remaining lives.

Code:

def print\_hangman(self):  
 print(self.hangman\_stages[self.lives])

# Step 4: Take Guess from Player

We write a function to handle user input:  
- Check if the letter was already guessed.  
- If correct, add it to guessed letters.  
- If wrong, decrease lives.

Code:

def guess\_letter(self, letter):  
 letter = letter.upper()  
 if letter in self.guessed\_letters:  
 print("You already guessed this letter!")  
 elif letter in self.word:  
 print(f" Good job! {letter} is in the word.")  
 self.guessed\_letters.append(letter)  
 else:  
 print(f" Wrong guess! {letter} is not in the word.")  
 self.guessed\_letters.append(letter)  
 self.lives -= 1

# Step 5: Check Win or Lose

We create functions to check if:  
- The player has guessed all letters (win).  
- The player has no lives left (lose).

Code:

def won(self):  
 return all(letter in self.guessed\_letters for letter in self.word)  
  
 def lost(self):  
 return self.lives <= 0

# Step 6: Run Game Loop

We create the run() function to:  
- Display the current state of the word.  
- Print hangman drawing.  
- Ask user to guess a letter.  
- Continue until win or lose condition is met.

Code:

def run(self):  
 print(" Welcome to HangmanGame ")  
 while not self.is\_won() and not self.is\_lost():  
 print("\nWord:", self.display\_word())  
 self.print\_hangman()  
 guess = input("Guess a letter: ").strip()  
   
 if len(guess) != 1 or not guess.isalpha():  
 print(" Please enter a single letter!")  
 continue  
  
 self.guess\_letter(guess)  
  
 if self.is\_won():  
 print("\n Congratulations! You guessed the word:", self.word)  
 else:  
 print("\n You lost! The word was:", self.word)  
 self.print\_hangman()

# Step 7: Run Game

Finally, we create an object of Hangman class and run the game using run() function.

Code:

game = HangmanGame("PYTHON")  
game.run()