Hangman Game in Python

Student Name: Ishmal Nadeem

Roll No: SU92-BSAIM-F24-101

Course: Artificial intelligence (Lab)

Semester 3B

Introduction

This project is a simple implementation of the classic Hangman game using Python. The player tries to guess the letters of a randomly chosen word. For each incorrect guess, one life is lost. The game continues until the player either guesses the word correctly or loses all lives.

Code Explanation

1. Importing Modules:

- `import random` is used to choose a random word from the list.
- `import hangman_stages` is used to show the Hangman ASCII art at each stage.

2. Word List and Lives:

- A predefined list of words is used: ['apple', 'banana', 'cherry', 'date', 'strawberry'].
- The player starts with 8 lives.

3. Choosing a Word:

- `random.choice(words)` randomly selects one word from the list.
- This is the secret word the player must guess.

4. Display Setup:

- A list of underscores `_` is created to represent unguessed letters.
- Example: For 'apple', it starts as ['_', '_', '_', '_', '_'].

5. Game Loop:

- The game runs in a `while not game_over` loop.
- Player inputs a guess, which is converted to lowercase.

- If the guessed letter is in the word, it replaces the corresponding underscores.
- If not, one life is deducted.

6. Winning and Losing Conditions:

- If all underscores are replaced with letters, the player wins.
- If lives reach 0, the player loses and the secret word is revealed.

7. Hangman Stages:

- After each guess, `hangman_stages.stages[lives]` prints the current stage of the Hangman drawing.

Why Each Part is Used

- `random.choice()` → to make the game unpredictable by choosing a random word.
- `for loop` → to check each position in the word and update guessed letters.
- `if guessed_letter not in word` → to check wrong guesses and reduce lives.
- `if '_' not in display` → to check if the player has won.
- `hangman_stages.stages` → to show a fun visual of the game progress.

Output Screenshots

- Starting the game

```
import random
import hangman_stages
List = ['apple', 'banana', 'cherry', 'date', 'strawberry']
lives = 8
choosen_word = random.choice(List)
print(choosen_word)


display = []
# for displaying dashes "_"
for i in range(len(choosen_word)):
    display += "_"
print(display)
```

- This code imports random module dor choosing random name from the list and help displaying dashes same as the number of letters in that name.

- Losing/Winning screen

```
game_over = False
while not game_over:
    guessed_letter = (input('Guess a letter: ')).lower()

    for position in range(len(choosen_word)):
        letter = choosen_word[position]
        if letter == guessed_letter:
            display[position] = guessed_letter
    print(display)

    if guessed_letter not in choosen_word:
        lives -= 1
        print(f"You guessed {guessed_letter}, that's not in the word. You lose a life.")
        if lives == 0:
            game_over = True
            print("You lose!!")
    if "_" not in display:
            game_over = True
            print("You win!!")
    print(f"You have {lives} lives left.")
    print(hangman_stages.stages[lives])
```

- This code includes loops (both for and while) , for loop is used for iteration over choosed letter over guessed letters , while loop is used for handling end of the game , if the game is over or not (winning and losing).
- The last 2 code lines help displaying total left lives , through text and hangman image.
-

Conclusion

This project helped in understanding the use of loops, conditionals, lists, and modules in Python. It also demonstrates how a simple game can be built using basic programming concepts.