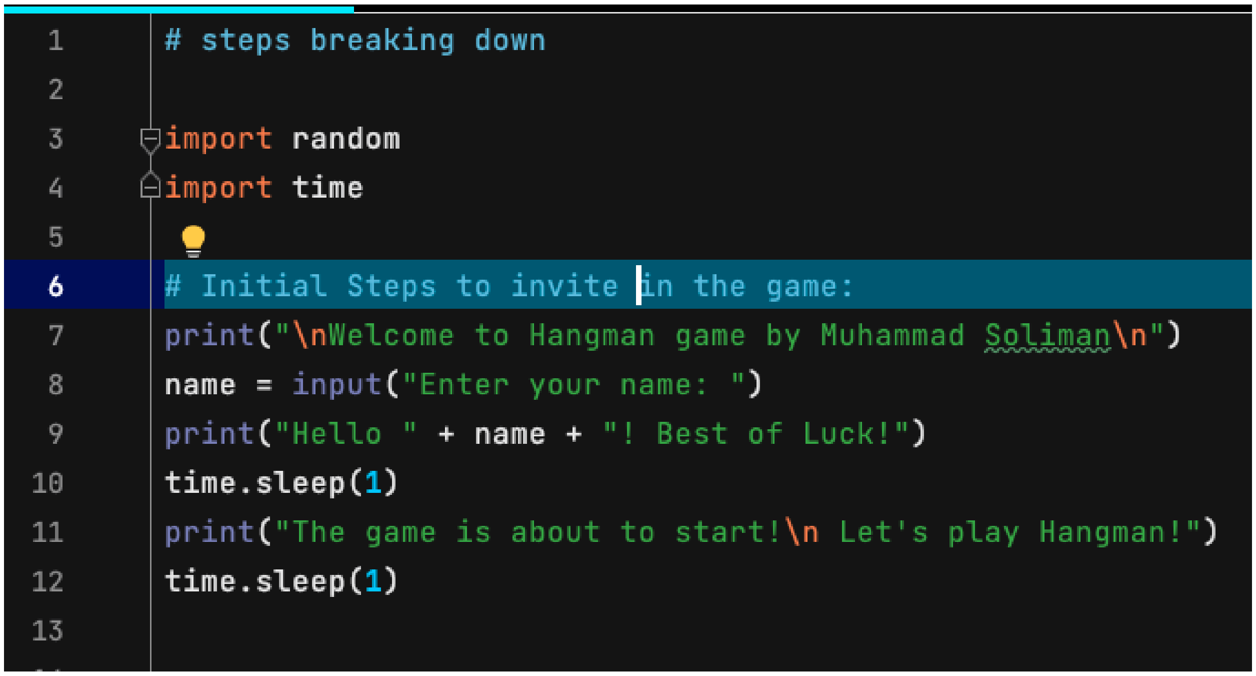
Project Title: Hangman

1. Introduction

Hangman is a classic word-guessing game in which players must guess the correct letters in a hidden word.
If the player guesses a letter that is not in the word, they lose one of their 6 chances until it shows part of the graph for the hangman.
If the player correctly guesses all of the letters in the word before running out of the 6 chances, they win the game.

2. Design and Implementation



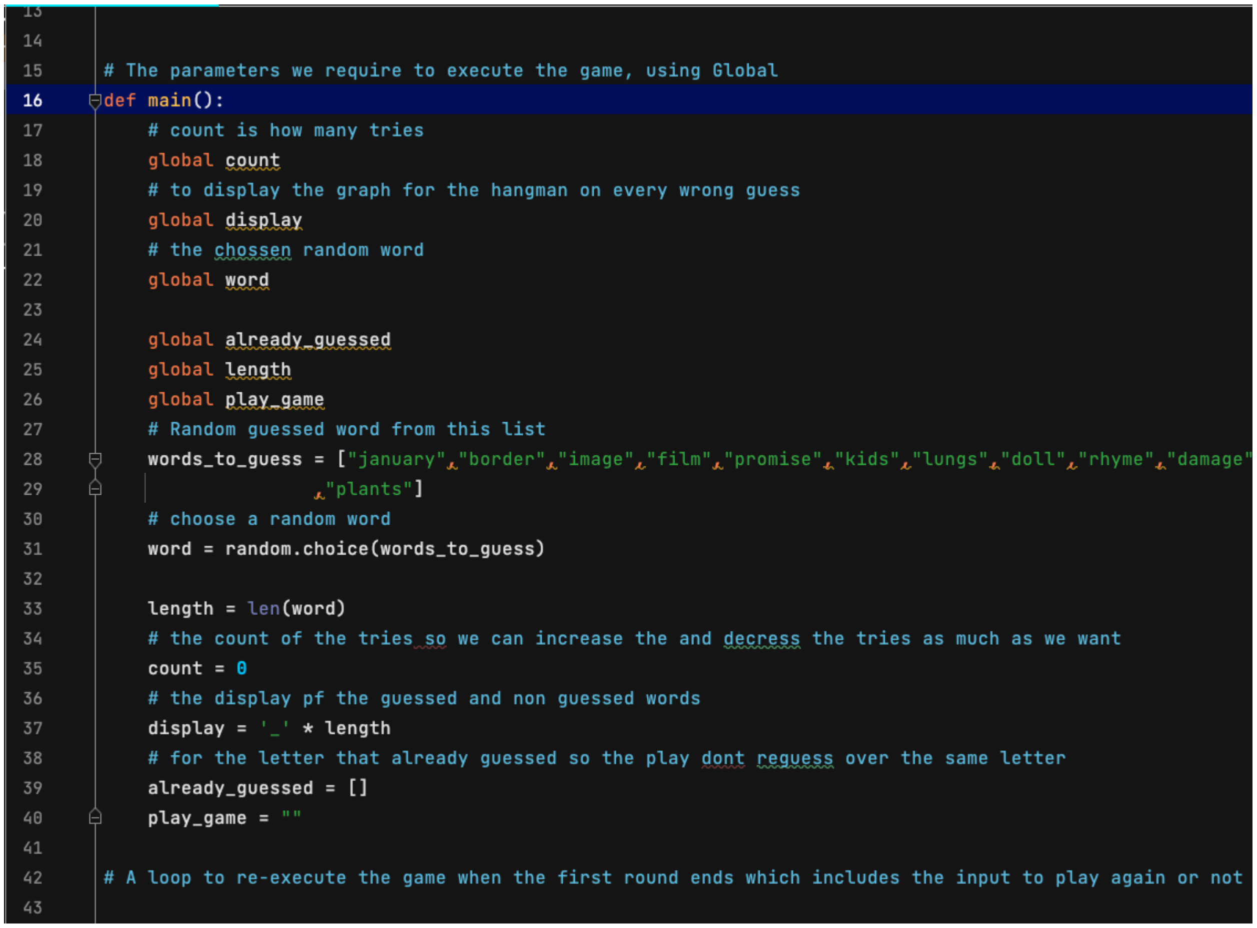
```python
# steps breaking down

import random
import time

# Initial Steps to invite in the game:
print("\nWelcome to Hangman game by Muhammad Soliman\n")
name = input("Enter your name: ")
print("Hello " + name + "! Best of Luck!")
time.sleep(1)
print("The game is about to start!\n Let's play Hangman!")
time.sleep(1)

```

- Give a detailed description of the design and implementation of your project.

  1- we need to import random where it will choose from a created list randomly

  2- Initial Steps to invite in the game where the play enters their name and welcoming message. I am using (time.sleep) to delaying showing the message to look more interesting

3- creating the parameter that is required to execute the game using the Global variable



```python

# The parameters we require to execute the game, using Global
def main():
    # count is how many tries
    global count
    # to display the graph for the hangman on every wrong guess
    global display
    # the chossen random word
    global word

    global already_guessed
    global length
    global play_game
    # Random guessed word from this list
    words_to_guess = ["january","border","image","film","promise","kids","lungs","doll","rhyme","damage"
                      ,"plants"]
    # choose a random word
    word = random.choice(words_to_guess)

    length = len(word)
    # the count of the tries so we can increase the and decress the tries as much as we want
    count = 0
    # the display pf the guessed and non guessed words
    display = '_' * length
    # for the letter that already guessed so the play dont reguess over the same letter
    already_guessed = []
    play_game = ""

# A loop to re-execute the game when the first round ends which includes the input to play again or not

```

At this step will start by creating a function <def main():>,. also, at this step, listing the words in a random module that will choose from for the game, at this step, I am using < word = random.choice(words_to_guess)>.

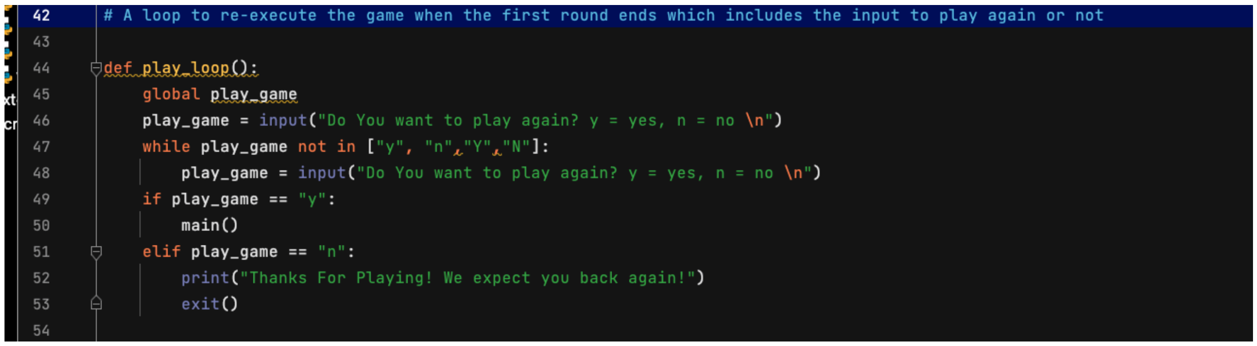
Global count: for number of the tries which will be limited to 6

Global Display: to show the hangman graph every time the player loses a point

Global already_guessed: the word that is already guessed by the player for not picking up the same word

<   length = len(word)>: for the words length

4- in this step, we will create a function for replay again when the player chooses to do so,



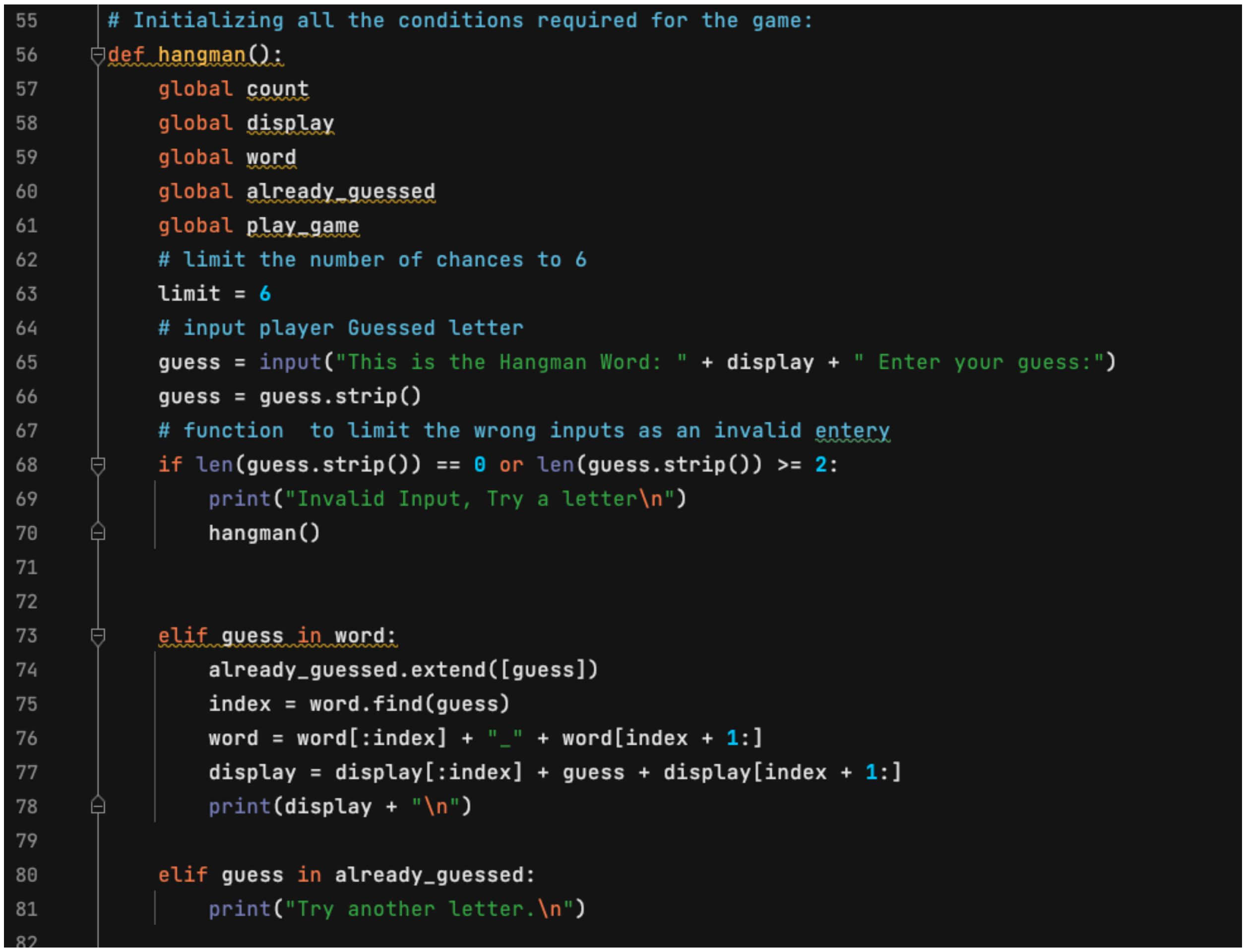
```python
# A loop to re-execute the game when the first round ends which includes the input to play again or not

def play_loop():
    global play_game
    play_game = input("Do You want to play again? y = yes, n = no \n")
    while play_game not in ["y", "n","Y","N"]:
        play_game = input("Do You want to play again? y = yes, n = no \n")
    if play_game == "y":
        main()
    elif play_game == "n":
        print("Thanks For Playing! We expect you back again!")
        exit()
```

which is a function we can call it when the player loses or win and chose to play again.

-First creat a def PLay_loop than add a <Global play_game>

- create an input of yes/no than a while loop to identify yes or no as a different choice. Where yes PLay again and start the main function and no Exit()

5- initializing all the conditions required for the game:

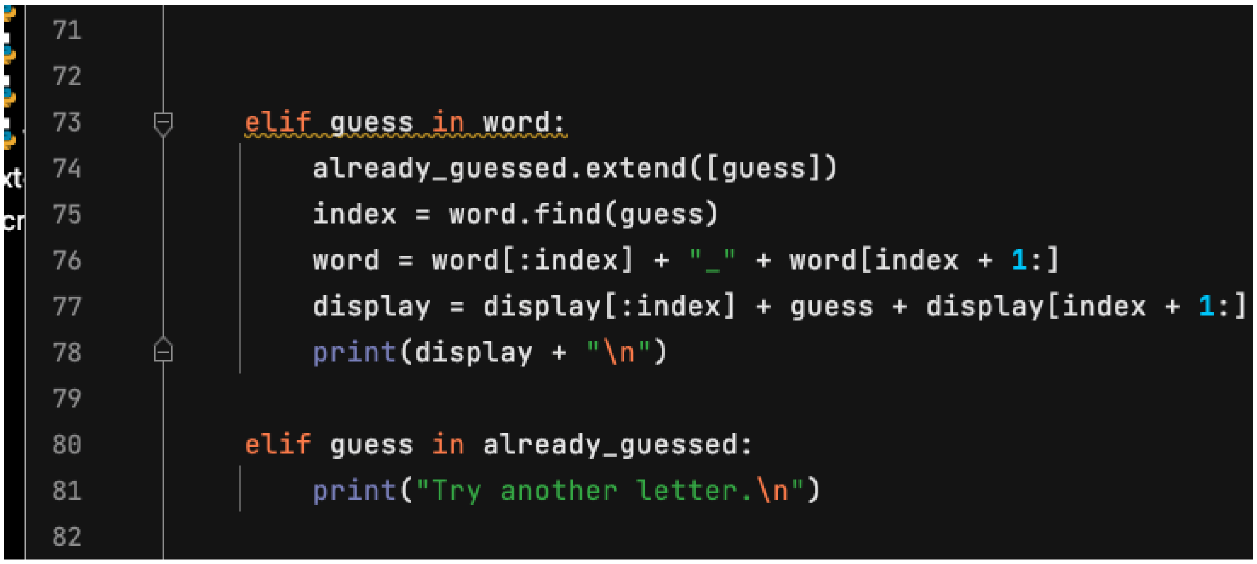


```python
# Initializing all the conditions required for the game:
def hangman():
    global count
    global display
    global word
    global already_guessed
    global play_game
    # limit the number of chances to 6
    limit = 6
    # input player Guessed letter
    guess = input("This is the Hangman Word: " + display + " Enter your guess:")
    guess = guess.strip()
    # function  to limit the wrong inputs as an invalid entery
    if len(guess.strip()) == 0 or len(guess.strip()) >= 2:
        print("Invalid Input, Try a letter\n")
        hangman()


    elif guess in word:
        already_guessed.extend([guess])
        index = word.find(guess)
        word = word[:index] + "_" + word[index + 1:]
        display = display[:index] + guess + display[index + 1:]
        print(display + "\n")

    elif guess in already_guessed:
        print("Try another letter.\n")
```

Create a function for the condition and rules of the game, <def hangman> we refer to some of the old global variables like <global count, global display, global word, global, already_guesse, global play_game. In addition, new variables will need it like Limit = 6 to limit the number of tries being wrong to 6 tries only. Using the strip() method removes leading and trailing whitespace characters from a string. This includes spaces, tabs, newline characters, and other whitespace characters.

-add IF statment to avoid wrong typing of letters like numbers instead of letters

This part code which consider the most complex one:



```
    elif guess in word:
        already_guessed.extend([guess])
        index = word.find(guess)
        word = word[:index] + "_" + word[index + 1:]
        display = display[:index] + guess + display[index + 1:]
        print(display + "\n")

    elif guess in already_guessed:
        print("Try another letter.\n")

```

- elif guess in word:

  already_guessed.extend([guess])

  index = word.find(guess)

  word = word[:index] + "_" + word[index + 1:]

  display = display[:index] + guess + display[index + 1:]

  print(display + "\n")

In this block of code, the `elif` statement is used to check if the player's guess (stored in the variable `guess`) is contained within the word that the player is trying to guess (stored in the variable `word`).

If the `elif` condition is true, then the code block is executed. Here's what each line does:

- `already_guessed.extend([guess])`: This line adds the player's guess to a list of already guessed letters (stored in the `already_guessed` list). This is done so that the player doesn't accidentally guess the same letter again.

- `index = word.find(guess)`: This line finds the index of the first occurrence of the player's guess within the word. For example, if the word is "apple" and the player guesses "p", then the index would be 1.

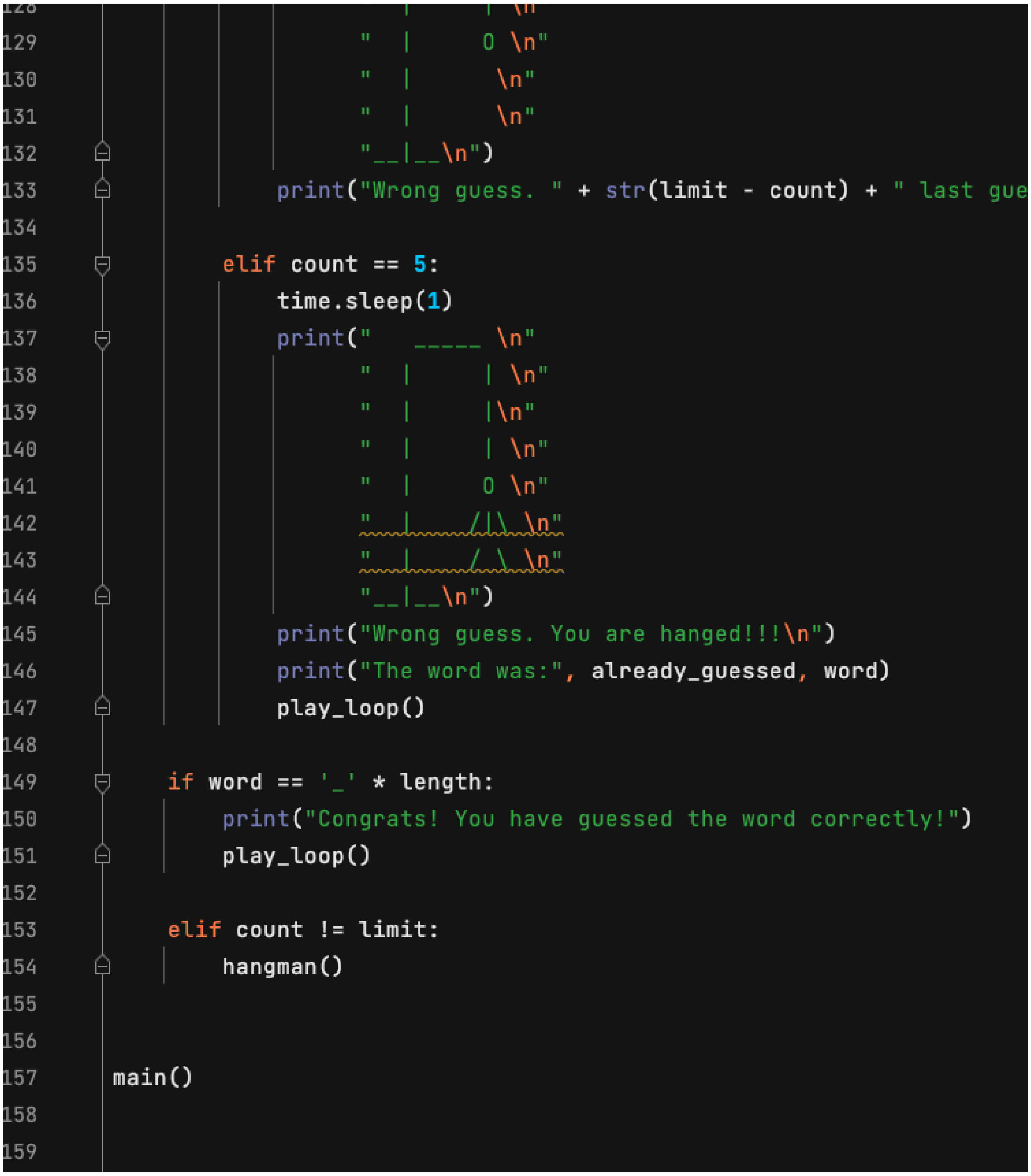
`word = word[:index] + "_" + word[index + 1:]`: This line replaces the correctly guessed letter in the word with an underscore. For example, if the word is "apple" and the player guesses "p", then the word would become "a_ple".

`display = display[:index] + guess + display[index + 1:]`: This line does the same thing as the previous line, but for the `display` variable. The `display` variable is likely used to show the player their progress in guessing the word.

- `print(display + "\n")`: This line prints the updated `display` variable, along with a newline character, to make the output easier to read.

6- At the end, the tries would be represented by showing part of the hangman once the players lose each time.

Then goes to the play loop after the game finish, plays again, or exit.



```
                  "  |     | \n"
                  "  |     O \n"
                  "  |      \n"
                  "  |      \n"
                  "__|__\n")
            print("Wrong guess. " + str(limit - count) + " last gue

        elif count == 5:
            time.sleep(1)
            print("   _____ \n"
                  "  |     | \n"
                  "  |     |\n"
                  "  |     | \n"
                  "  |     O \n"
                  "  |    /|\ \n"
                  "  |    / \ \n"
                  "__|__\n")
            print("Wrong guess. You are hanged!!!\n")
            print("The word was:", already_guessed, word)
            play_loop()

    if word == '_' * length:
        print("Congrats! You have guessed the word correctly!")
        play_loop()

    elif count != limit:
        hangman()


main()
```

- In particular, this section should contain:

    - Details of how you converted from design to the actual realization of your project in terms of implementing the code.
    - Any choices that you made, and any modifications that you made to the design, in response to difficulties that you might have encountered while implementing the project.
- Include relevant screenshots of your game at different stages of play in the report.

3. Conclusions

- Discuss what you personally learned from your project.

  I learned some new stuff like global, which I found out later is not recommended to use because it creates confusion . also, the time module is new for me, which is exciting.

  Getting the projet together that we do every week teach me the most Sorry Andrew, the project so far is the best teacher to implment your teaching.