DEPARTMENT OF COMPUTER & INFORMATION SYSTEMS ENGINEERING
BACHELORS IN COMPUTER SYSTEMS ENGINEERING
Course Code: CS-115
Course Title: Computer Programming
Complex Engineering Problem
FE Batch 2022, Fall Semester 2022
Grading Rubric
TERM PROJECT

Group members:

Student No	Name	Roll No
S1	ZAID AHAMED	CS - 154
S2	MUHAMMED	CS - 155
S3	AMNA	CS - 156

CRITERIA AND SCALES				**Marks Obtained**		
				S1	**S2**	**S3**
Criterion 1: Does the application meet the desired specifications and produce the desired outputs? (CPA-1, CPA-3) **[8 marks]**						
1	2	3	4			
The application does not meet the desired specifications and is producing incorrect outputs.	The application partially meets the desired specifications and is producing incorrect or partially correct outputs.	The application meets the desired specifications but is producing incorrect or partially correct outputs.	The application meets all the desired specifications and is producing correct outputs.			
Criterion 2: How well is the code organization? **[2 marks]**						
1	2	3	4			
The code is poorly organized and very difficult to read.	The code is readable only to someone who knows what it is supposed to be doing.	Some part of the code is well organized, while some part is difficult to follow.	The code is well organized and very easy to follow.			
Criterion 3: How friendly is the application interface? (CPA-1, CPA-3) **[2 marks]**						
1	2	3	4			
The application interface is difficult to understand and use.	The application interface is easy to understand and but not that comfortable to use.	The application interface is very easy to understand and use.	The application interface is very interesting/ innovative and easy to understand and use.			
Criterion 4: How does the student performed individually and as a team member? (CPA-2, CPA-3) **[4 marks]**						
1	2	3	4			
The student did not work on the assigned task.	The student worked on the assigned task, and accomplished goals partially.	The student worked on the assigned task, and accomplished goals satisfactorily.	The student worked on the assigned task, and accomplished goals beyond expectations.			
Criterion 5: Does the report adhere to the given format and requirements? **[4 marks]**						
1	2	3	4			
The report does not contain the required information and is formatted poorly.	The report contains the required information only partially but is formatted well.	The report contains all the required information but is formatted poorly.	The report contains all the required information and completely adheres to the given format.			
			Total Marks:			

Teacher's Signature

CONTENTS

PROBLEM DESCRIPTION

Introduction to Hangman game

The hangman game in python is written in a python programming language, In this Hangman Game Project is to implement the Game Using Python. It doesn’t require any specific modules other than random. Python loops and functions are enough to build this game here.

A hangman game on python is about guessing letters (A-Z) to form the words. If the player guesses the right letter that is within the word, the letter appears at its correct position. The user has to guess the correct word until Guess becomes 0 , if then the game is over.

Our application will allow the user to play the classic word game Hangman against the computer. Our application maintains two interfaces: one for the player(Main interface) and one for the administrator, as shown in the following diagram. For the game, the computer picks a word, randomly form a list of available words, and the player tries to guess letters in the word. The player is given a certain number of guesses at the beginning. The game is interactive; as the player inputs his/her guess, the computer either: reveals the letter if it exists in the secret word or penalize the user and updates the number of guesses remaining. The game ends when either the user guesses the secret word, or the user runs out of guesses.

DISTINGUISHING FEATURES OF YOUR PROJECT

- Keeps track of the players who played the game before and it can be erased by the admin.
- For interactions we used emojis for displaying.

GAME RULES

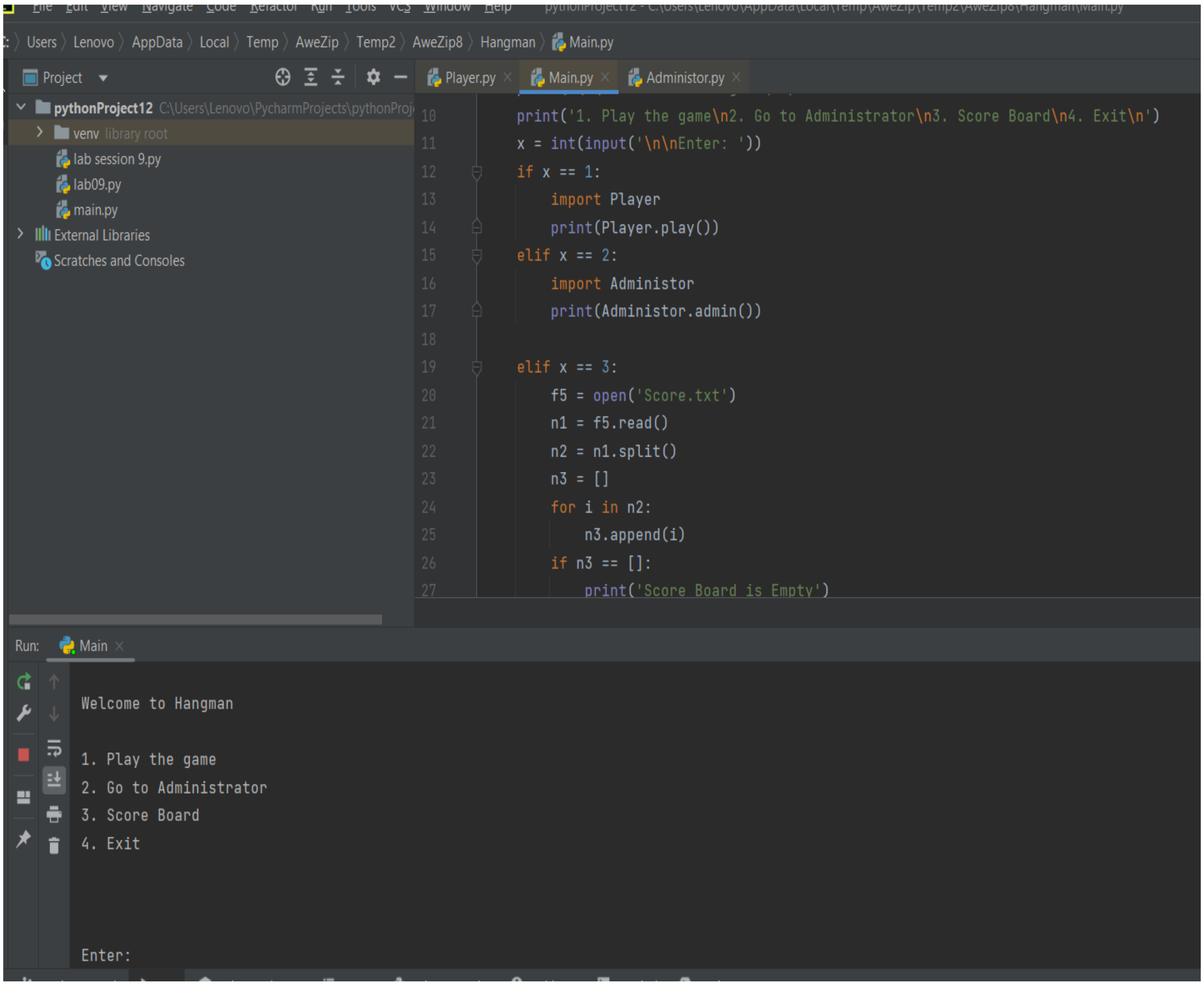
- The computer must select a random word from the list.
- Users start with 6 guesses and 3 warnings.
- At the start of the game, let the player know how many letters in the secret word and how many guesses and warnings left.
- If the player has not guessed so far and run terminal shows the player the remaining letters before each turn he/she types.
- Ask the player to supply one guess at a time. Display to the player the secret word, with guessed letters displayed and un-guessed letters replaced with an underscore and space (_).
- The game accepts both upper- and lower-case letters as valid guesses. If the player inputs anything other than alphabets, prompt the user to enter valid input.
- If the player inputs a letter that hasn't been guessed before and the letter is in the secret word, the player does not lose any guesses or warnings.
- If the player inputs a consonant and the consonant is not in the secret word, the user loses one guess.
- If the vowel has been guessed and the vowel is not in the secret word, the player loses two guesses.
- Each time the player inputs anything besides an alphabet (symbols, numbers) or a letter that has already been guessed, the player loses a warning. If no warnings are left, the player loses a guess.
- The game should end when the player constructs the full word or runs out of guesses.
- If the player runs out of guesses before completing the word, tell them that the game has been lost and reveal the word. The game ends.
- If the player wins print a congratulatory message and tell the player the score calculated as follows: Total score = number of guesses remaining x number unique letters in the secret word
- The game must also keep track of the highest score along with the name of the player and displays a special message if the player achieves a new high score.

INTERFACES

There are 2 interfaces in the hangman game.

1. MAIN INTERFACE

It provides the play option, Administrator interface, scoreboard (which shows the player’s score) and exit using a while loop.

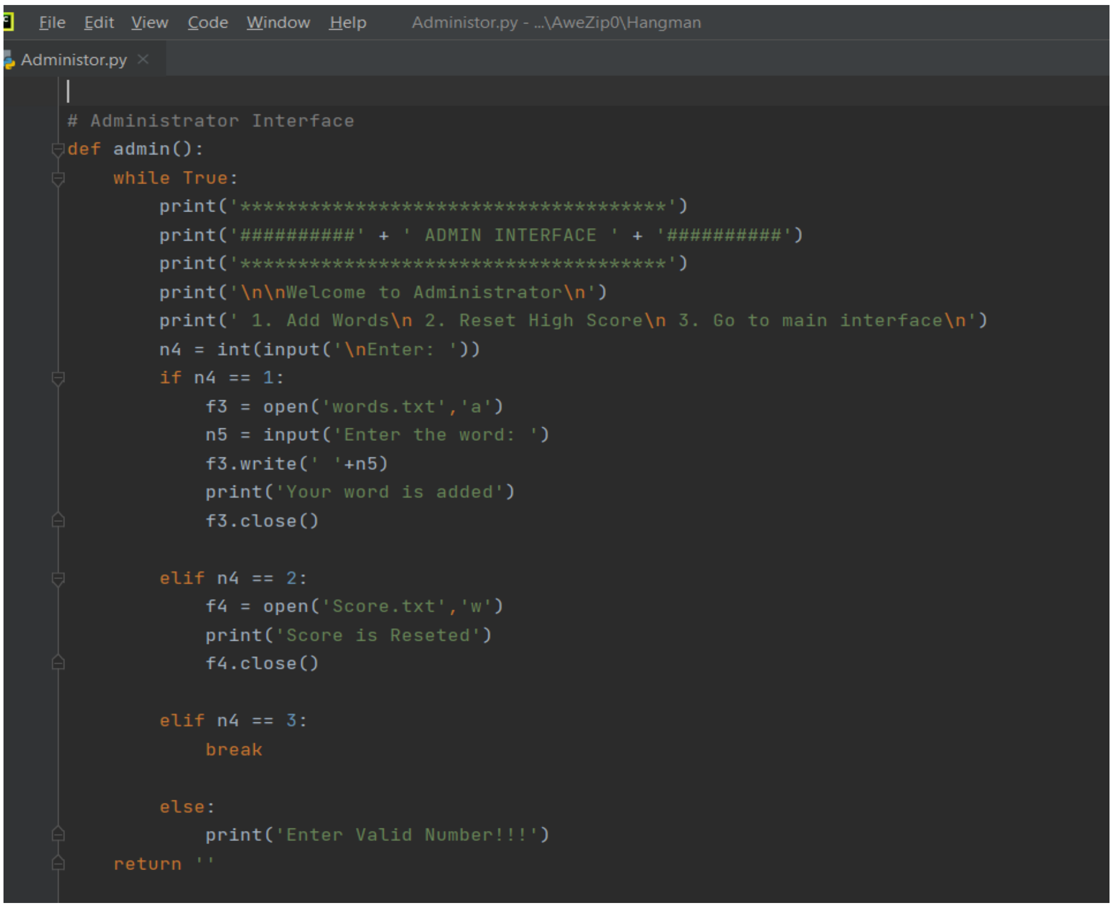


```
print('1. Play the game\n2. Go to Administrator\n3. Score Board\n4. Exit\n')
x = int(input('\n\nEnter: '))
if x == 1:
    import Player
    print(Player.play())
elif x == 2:
    import Administor
    print(Administor.admin())

elif x == 3:
    f5 = open('Score.txt')
    n1 = f5.read()
    n2 = n1.split()
    n3 = []
    for i in n2:
        n3.append(i)
    if n3 == []:
        print('Score Board is Empty')
```

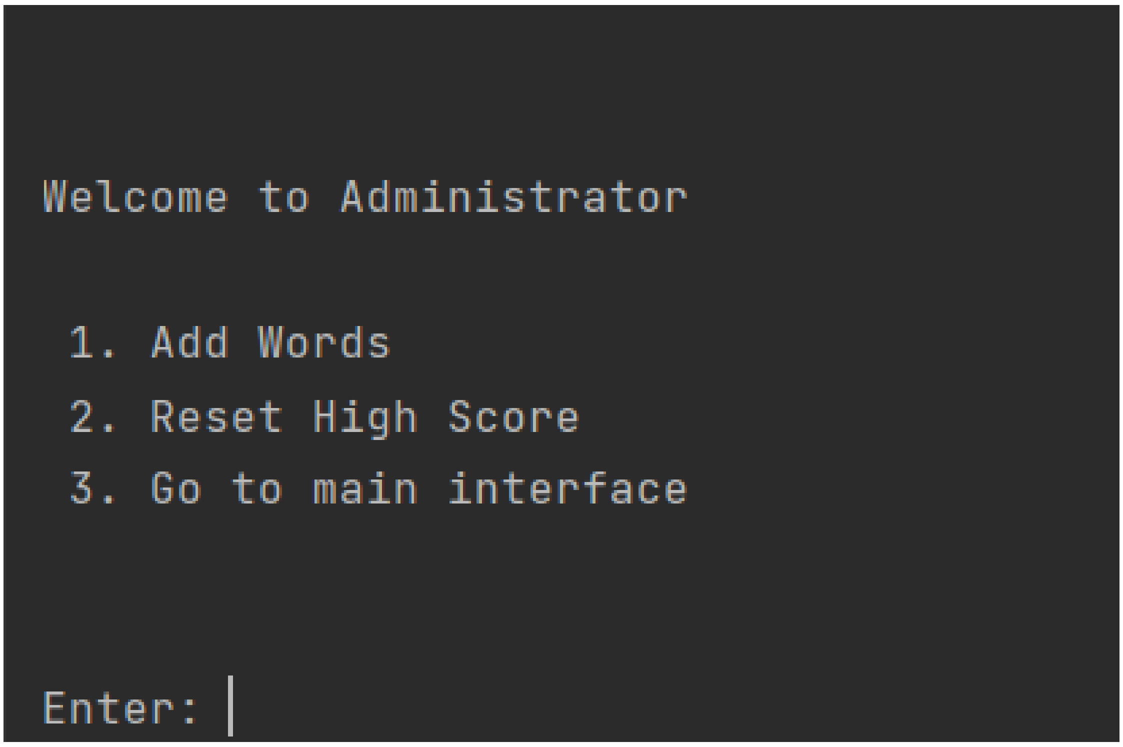
```
Welcome to Hangman

1. Play the game
2. Go to Administrator
3. Score Board
4. Exit



Enter:
```

2. ADMINISTRATOR INTERFACE

In the administrator interface we can add words, reset high score and it also provides option to return to the main interface too. Here we were using, **while loop**, **defined function,** and **if statement.**


File Edit View Code Window Help Administor.py - ...\AweZip0\Hangman
Administor.py

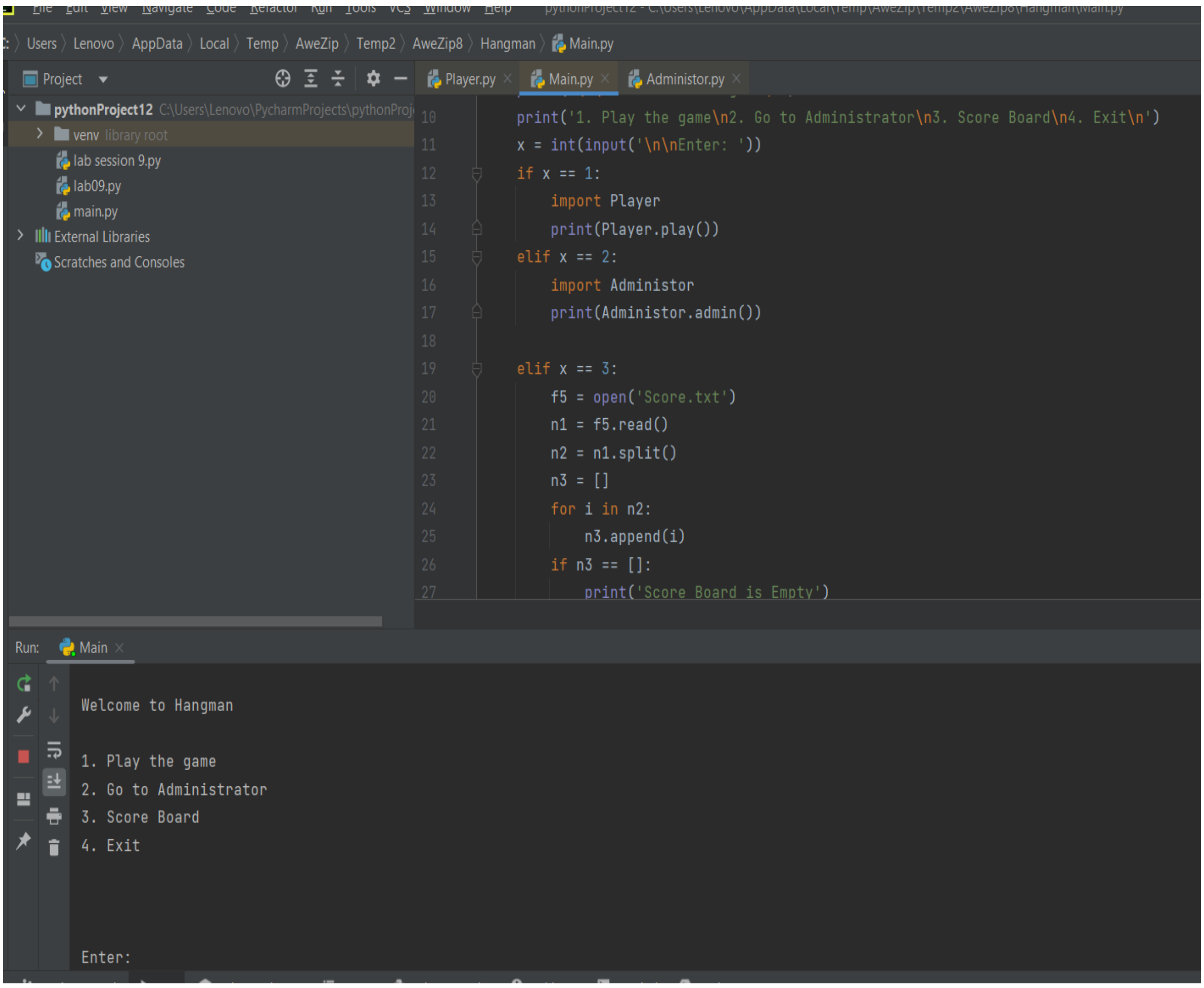
```python
# Administrator Interface
def admin():
    while True:
        print('*************************************')
        print('##########' + ' ADMIN INTERFACE ' + '##########')
        print('*************************************')
        print('\n\nWelcome to Administrator\n')
        print(' 1. Add Words\n 2. Reset High Score\n 3. Go to main interface\n')
        n4 = int(input('\nEnter: '))
        if n4 == 1:
            f3 = open('words.txt','a')
            n5 = input('Enter the word: ')
            f3.write(' '+n5)
            print('Your word is added')
            f3.close()

        elif n4 == 2:
            f4 = open('Score.txt','w')
            print('Score is Reseted')
            f4.close()

        elif n4 == 3:
            break

        else:
            print('Enter Valid Number!!!')
    return ''
```



```
Welcome to Administrator

 1. Add Words
 2. Reset High Score
 3. Go to main interface


Enter: 
```

FLOW OF OUR PROJECT

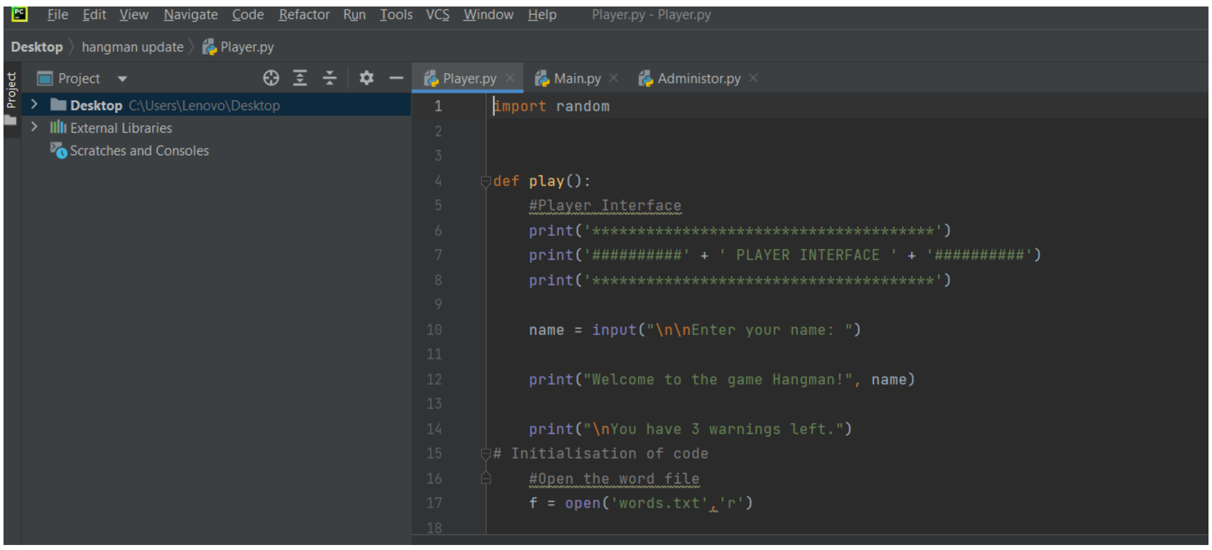
❖ The Game starts in the Main Interface.



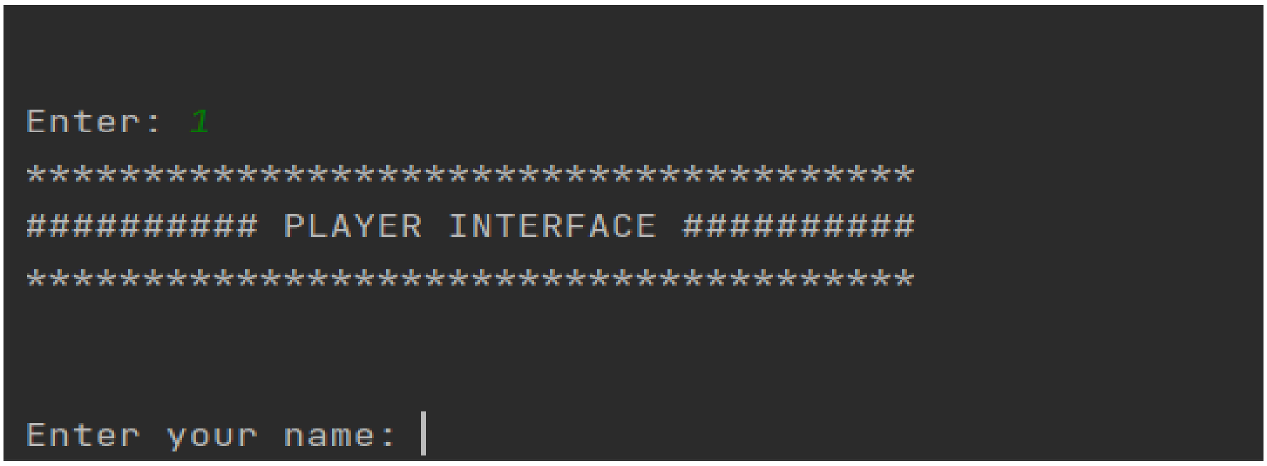
```
print('1. Play the game\n2. Go to Administrator\n3. Score Board\n4. Exit\n')
x = int(input('\n\nEnter: '))
if x == 1:
    import Player
    print(Player.play())
elif x == 2:
    import Administor
    print(Administor.admin())

elif x == 3:
    f5 = open('Score.txt')
    n1 = f5.read()
    n2 = n1.split()
    n3 = []
    for i in n2:
        n3.append(i)
    if n3 == []:
        print('Score Board is Empty')
```

```
Welcome to Hangman

1. Play the game
2. Go to Administrator
3. Score Board
4. Exit



Enter:
```

❖ If the user wants to play the game user should enter the number 1.

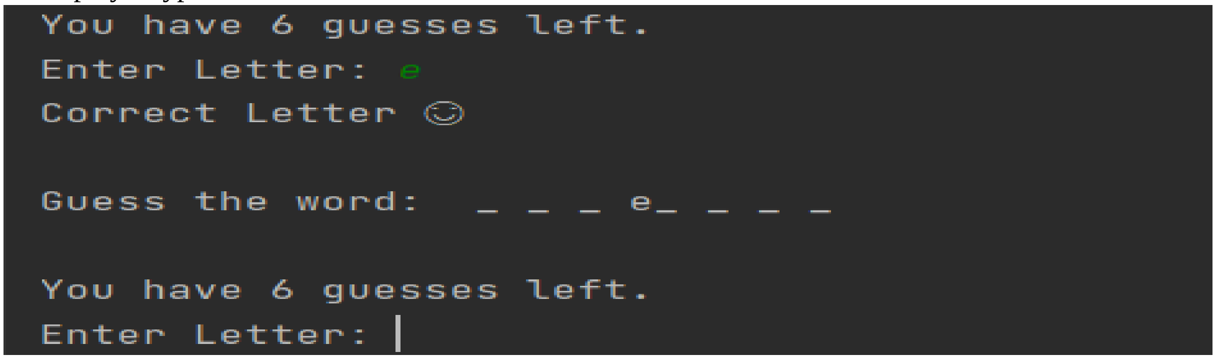


```
import random


def play():
    #Player Interface
    print('***************************************')
    print('##########' + ' PLAYER INTERFACE ' + '##########')
    print('***************************************')

    name = input("\n\nEnter your name: ")

    print("Welcome to the game Hangman!", name)

    print("\nYou have 3 warnings left.")
# Initialisation of code
    #Open the word file
    f = open('words.txt','r')
```



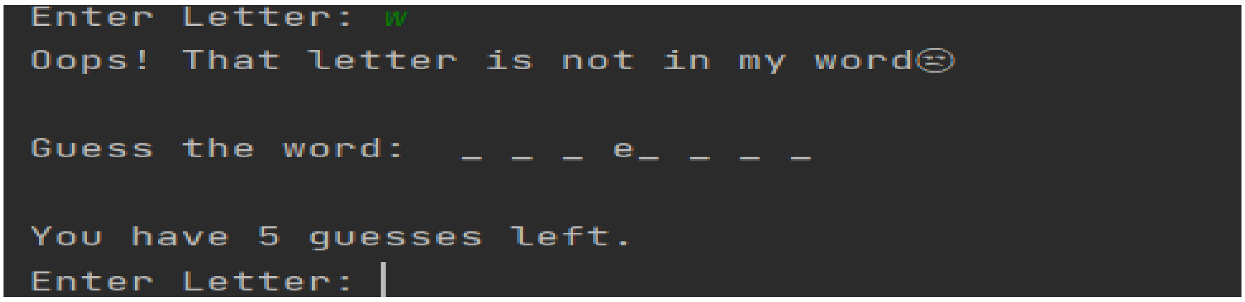
```
Enter: 1
**************************************
########## PLAYER INTERFACE ##########
**************************************


Enter your name:
```

❖ If the player types a correct letter which is in the word.



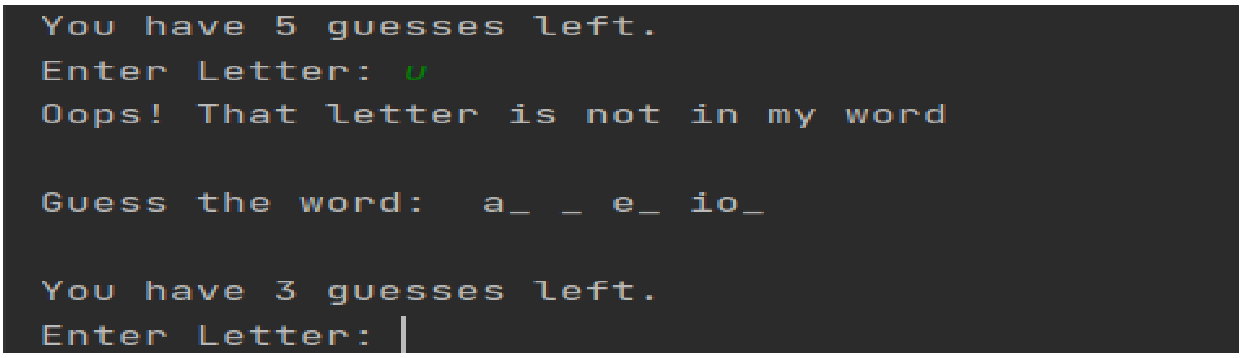
```
You have 6 guesses left.
Enter Letter: e
Correct Letter ☺

Guess the word:  _ _ _ e_ _ _ _

You have 6 guesses left.
Enter Letter:
```

❖ If the player enters a wrong letter that is not in the word.



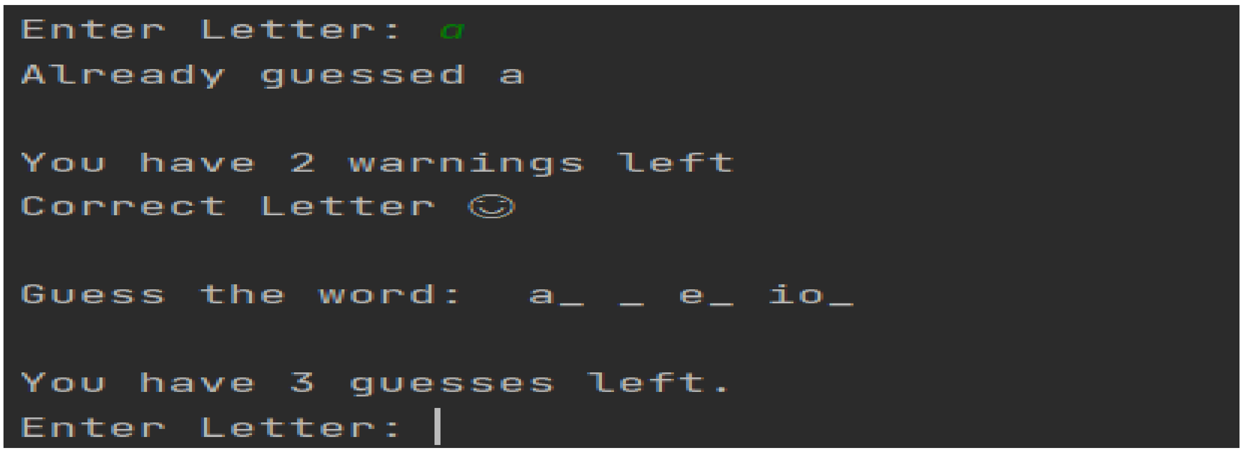
```
Enter Letter: w
Oops! That letter is not in my word☹

Guess the word:  _ _ _ e_ _ _ _

You have 5 guesses left.
Enter Letter:
```

❖ If we enter a wrong vowel in the game 2 guesses will be deducted.



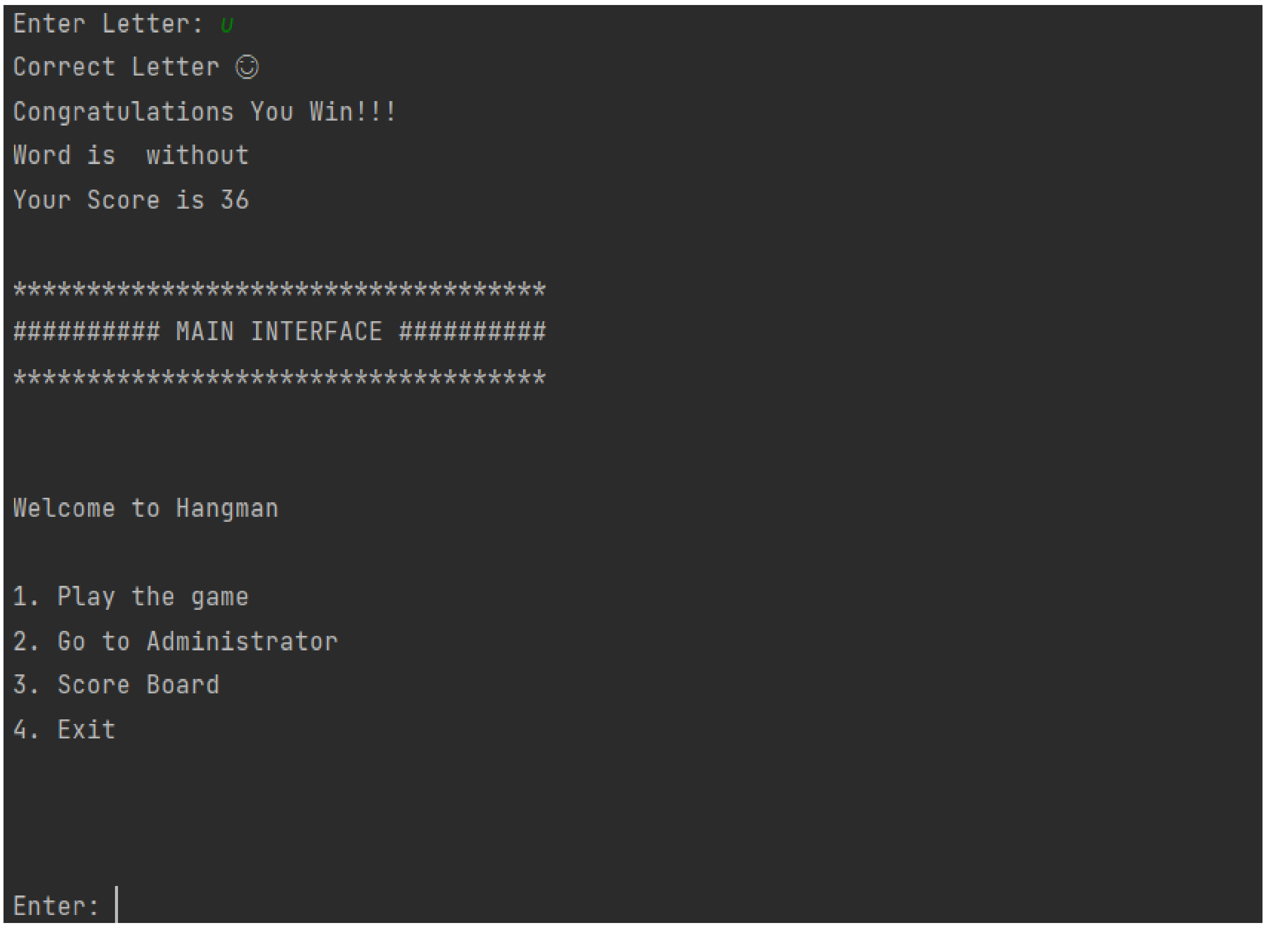
```
You have 5 guesses left.
Enter Letter: u
Oops! That letter is not in my word

Guess the word:  a_ _ e_ io_

You have 3 guesses left.
Enter Letter:
```

- If we enter a letter that was entered before a warning will be deducted.



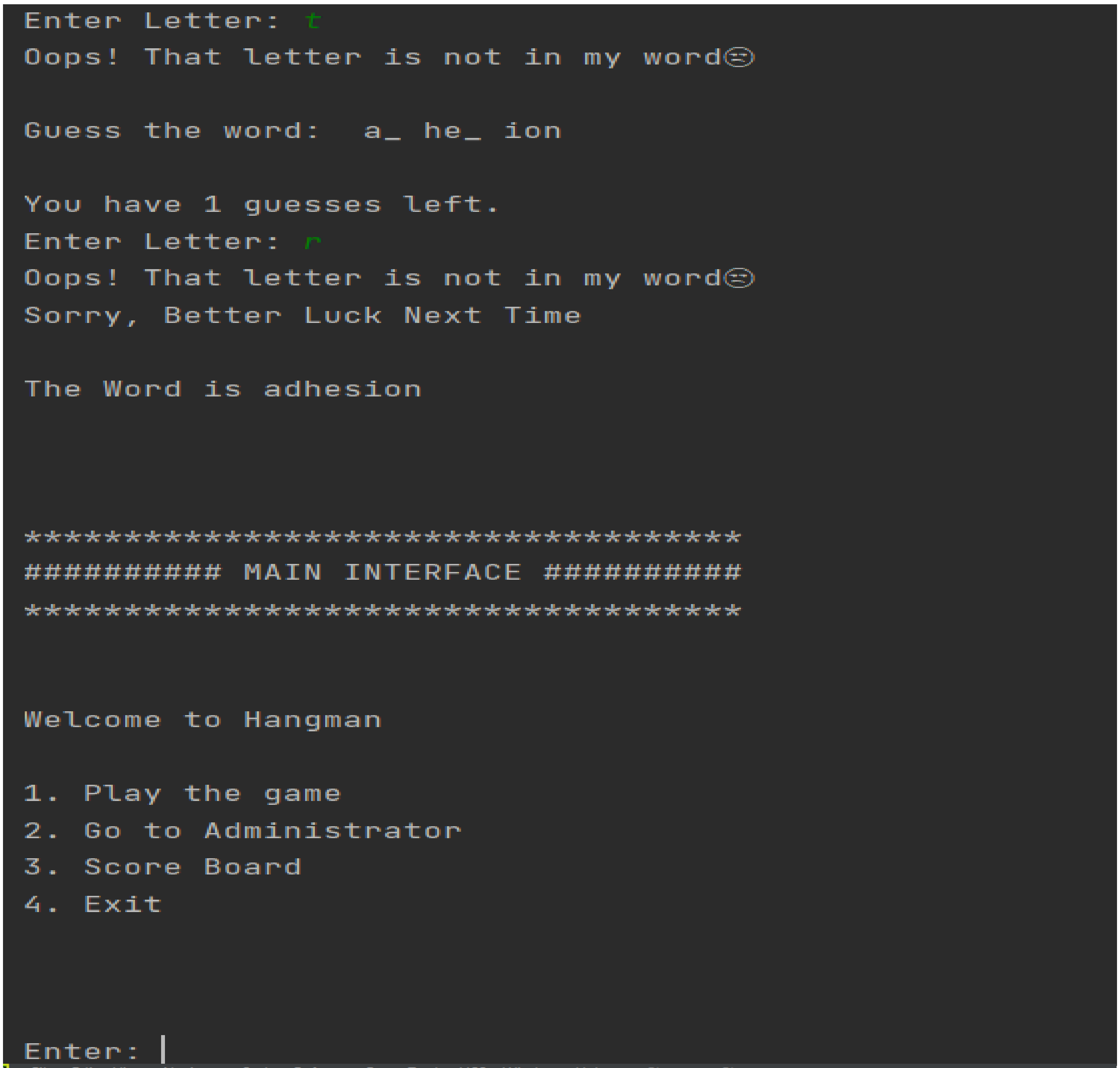
```
Enter Letter: a
Already guessed a

You have 2 warnings left
Correct Letter ☺

Guess the word:  a_ _ e_ io_

You have 3 guesses left.
Enter Letter:
```

- If the game ends (if the player wins or loses), again the player is redirected to the Main interface. When Winning,

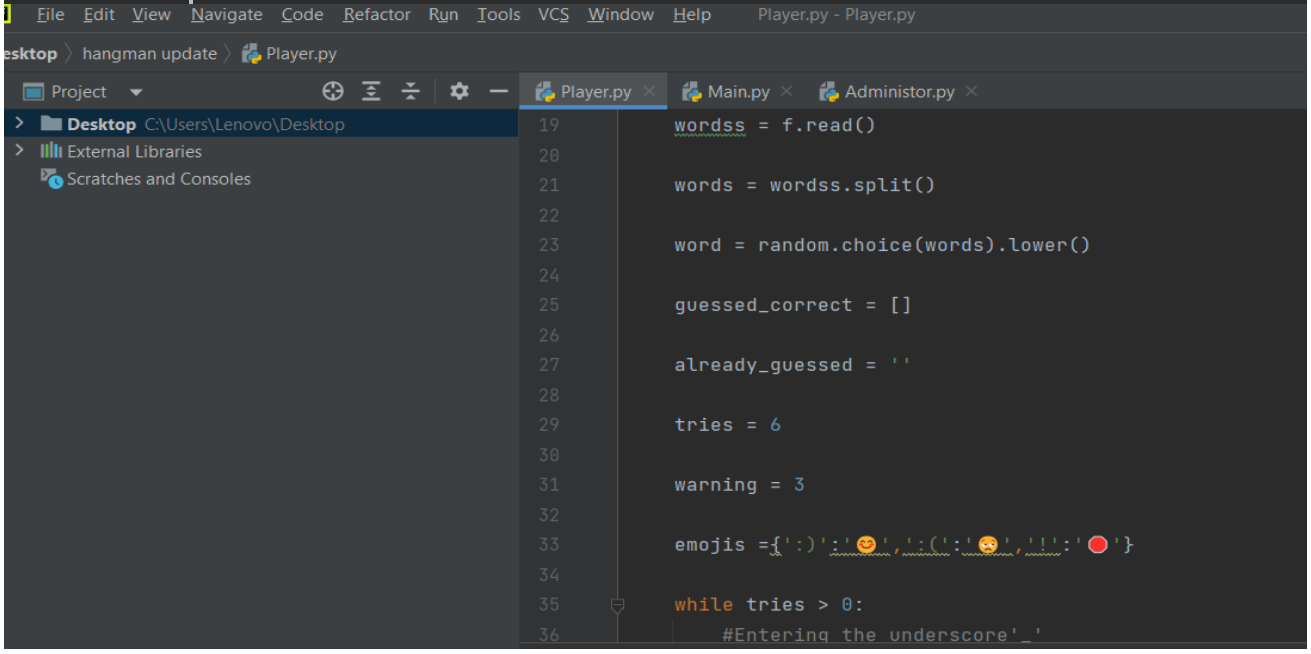


```
Enter Letter: u
Correct Letter ☺
Congratulations You Win!!!
Word is  without
Your Score is 36

************************************
########## MAIN INTERFACE ##########
************************************



Welcome to Hangman

1. Play the game
2. Go to Administrator
3. Score Board
4. Exit




Enter:
```

When losing,



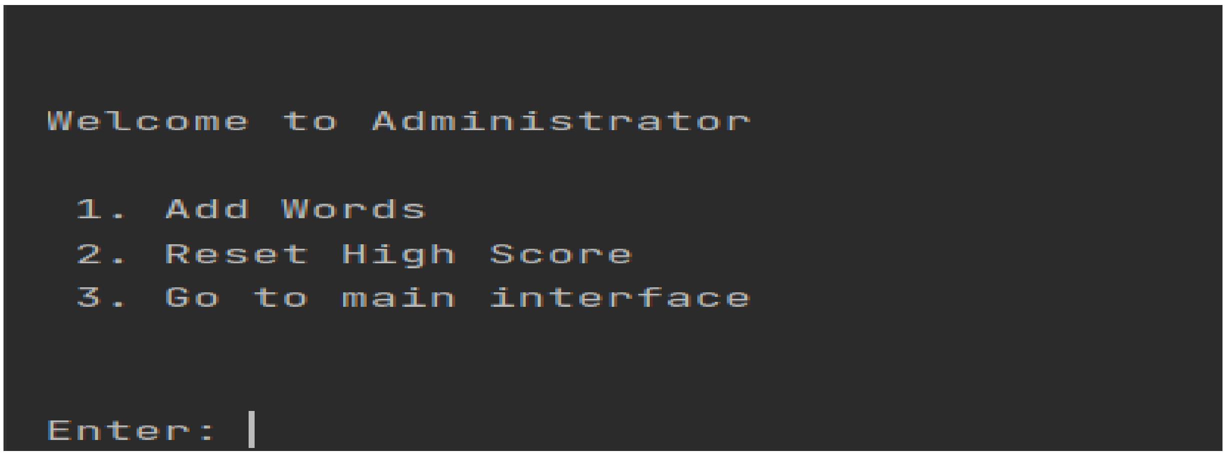
```
Enter Letter: t
Oops! That letter is not in my word😑

Guess the word:  a_ he_ ion

You have 1 guesses left.
Enter Letter: r
Oops! That letter is not in my word😑
Sorry, Better Luck Next Time

The Word is adhesion



************************************
########## MAIN INTERFACE ##########
************************************


Welcome to Hangman

1. Play the game
2. Go to Administrator
3. Score Board
4. Exit



Enter: 
```



File Edit View Navigate Code Refactor Run Tools VCS Window Help Player.py - Player.py
esktop 〉 hangman update 〉 Player.py
Project
Desktop C:\Users\Lenovo\Desktop
External Libraries
Scratches and Consoles
Player.py
Main.py
Administor.py

```python
        wordss = f.read()

        words = wordss.split()

        word = random.choice(words).lower()

        guessed_correct = []

        already_guessed = ''

        tries = 6

        warning = 3

        emojis ={':)':'😊',':(':'😒','!':'🔴'}

        while tries > 0:
            #Entering the underscore'_'
```

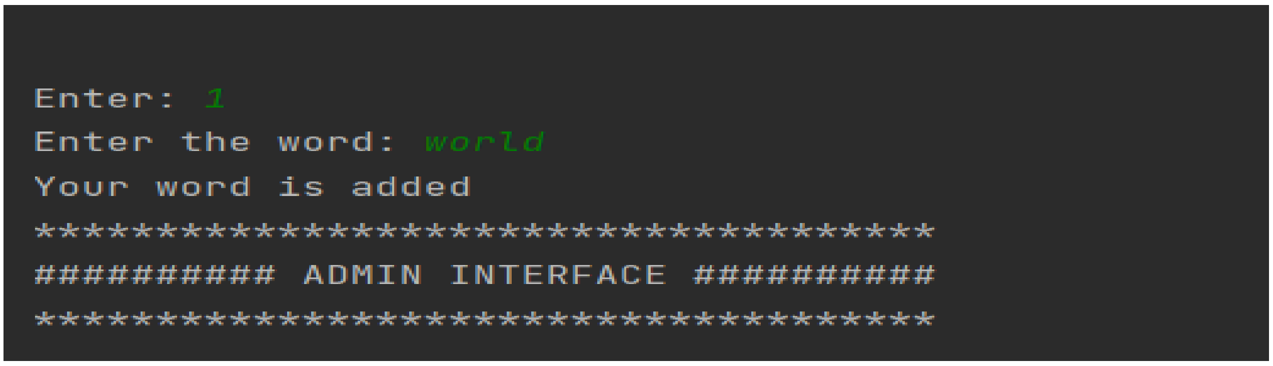
- If the user wants to go Administrator interface, user wants to type number 2.



```
 Welcome to Administrator

   1. Add Words
   2. Reset High Score
   3. Go to main interface


 Enter: 
```

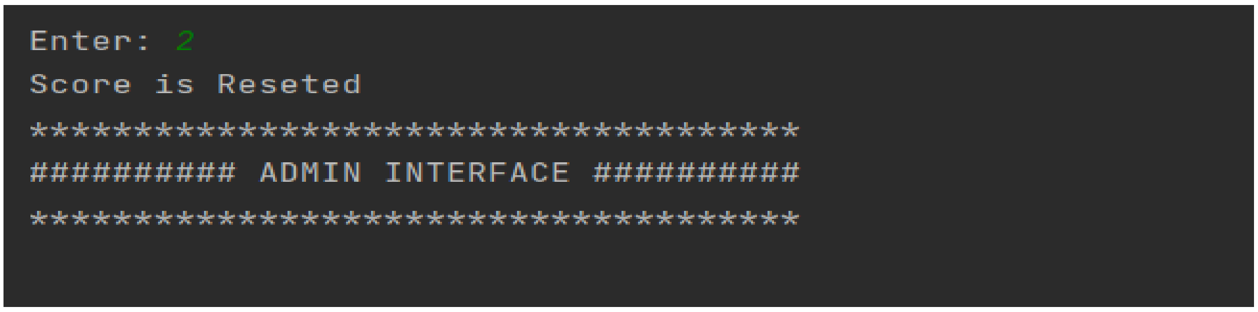
- In the Administrator interface user can,

1. Add the word (If the user enters number 1)



```
Enter: 1
Enter the word: world
Your word is added
*************************************
########## ADMIN INTERFACE ##########
*************************************
```

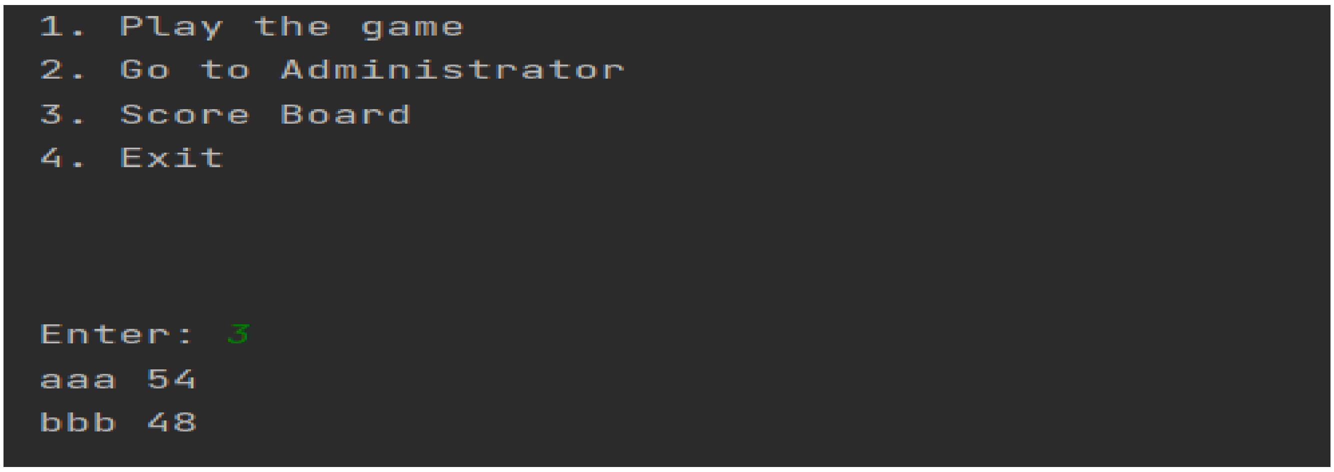
2. Reset the High Score (If user enters number 2)



```
Enter: 2
Score is Reseted
*************************************
########## ADMIN INTERFACE ##########
*************************************
```

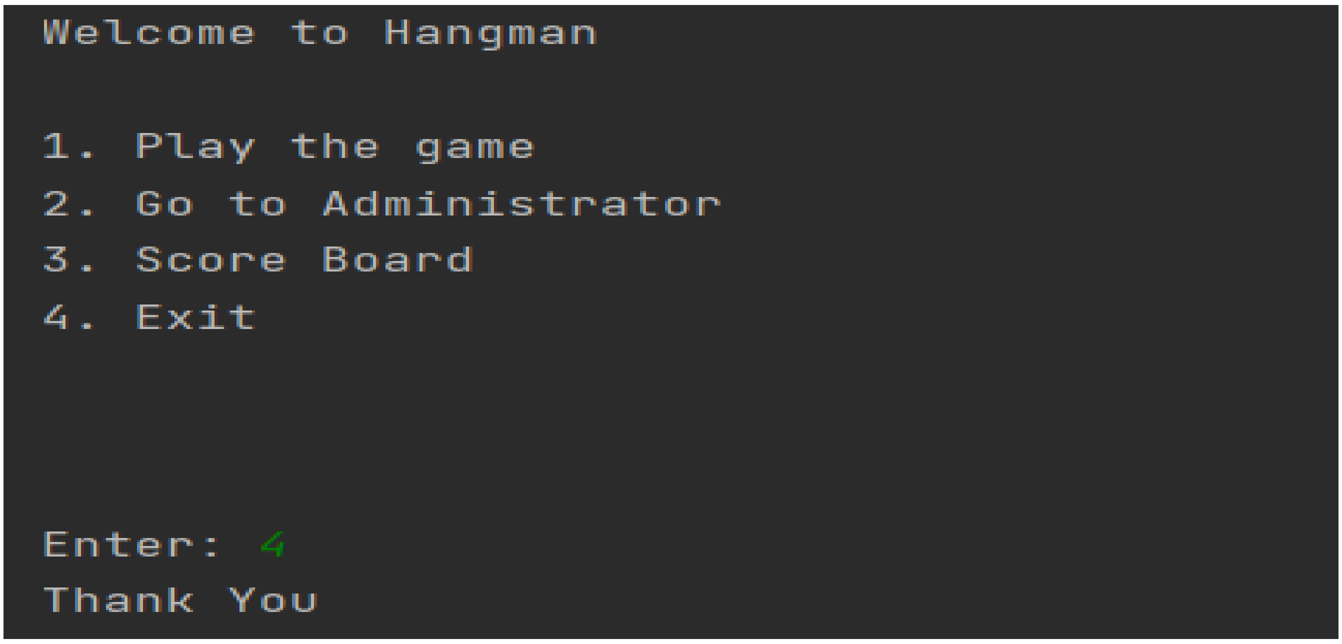
3. Go to the Main interface (If user enters number 3)

- If the user wants to look at the Scoreboard, the user should enter number 3.



```
1. Play the game
2. Go to Administrator
3. Score Board
4. Exit



Enter: 3
aaa 54
bbb 48
```

- If the user wants to Exit the Game user wants to enter number 4.



```
Welcome to Hangman

1. Play the game
2. Go to Administrator
3. Score Board
4. Exit




Enter: 4
Thank You
```

CHALLENGING PART WHILE WORKING ON THE PROJECT

- We faced some errors While reading the scoreboard file while the scoreboard is empty.
- We faced a problem While inserting an underscore with space (‘_ ’).
- We faced a problem While printing high scores if a player gets a high score.
- We had issues when importing the xxxx.py files.

ANY NEW THING LEARNT IN PYTHON WHILE WORKING ON THE PROJECT

- We learned to tackle with filing concept and understood some methods of filing.
- We learned about function and how we can use one function at many times.
- We learned about modules like random.
- We learned about how a while loop will work inside the function.
- We learned about how we can manipulate many conditions inside the function like while loop.

INDIVIDUAL CONTRIBUTIONS OF EACH GROUP MEMBER IN THE PROJECT

- ZAID AHAMED (CS - 154)
  - Filing and conditions of code

- MUHAMMED (CS - 155)
  - Player code

- AMNA (CS - 156)
  - Main interface and Administrator interface

FUTURE EXPANSIONS

- We are planning to add GUI for interfaces, in running the game, for pause and resume options, and displaying a man hanging sequentially when he/she types the wrong letter.
- We are planning to add password options so we can protect administrator mode from normal user.

CASE RUNS

- Included in the game flow