Foundation Certificate in Higher Education

Module: DOC334-Computer Programming

Module Leader: Mr. Nishanan Saliya Harankahawa

Assignment Type: Individual Coursework

Submission date: 24th December 2023

Group: E

Student ID	Student Name
20230297	**Mr. Anusigan Sivananthan**

I. Acknowledgment

I would like to extend my heartfelt thanks to my DOC334 Computer programming lecturer and module leader Mr.Nishan saliya sir who gave guidance and support to complete the individual coursework with excellence. I would also like to thank my friends and seniors who helped me to clear the doubts regarding the coursework.

II. Table of Contents

III. List of Tables

IV. List of Figures

1. Problem Statement

Aim of this project to develop a single player text based game “DON (Destroyer of Numbers)”. The player should defend “letter-kind” from evil numbers. DON can achieve it by selecting a lower numbers than his Life score throughout all 20 attempts to win the game. At start the program should ask for the name of the player and assign it to DON. Every game session should start with a randomly generated Life score between 1-50 as the initial life score of the player. Then Don encounters five random numbers generated to fight with across all 20 attempts. These randomly generated fighting numbers should be in range between 15-100 for 1-5 attempts, it should be in range between 250-2000 for 6-10 attempts , it should be in range between 3000-10000 for 11-15 attempts and it should be in range between 20000-100000 for 16-20 attempts.

If the player selects a number lower than or equal to DON’s Life score he will kill that number, adding its value to his Life score and allowing him to proceed to next attempt. If the player selects a number greater than DON’s Life score the number will kill DON and the game should end by displaying the game status. Inputting an invalid number not given in numbers to fight will also terminates the game and display the game status. Player who survives through all 20 attempts will be declared as winner and game status will be displayed. The player does not have an opportunity to quit the game until he select a number higher than his life score or he has completed all the 20 attempts successfully or enter any number which is not present in numbers to fight.

The game statistics should display the player name, Final score, Total attempts, status. Once the game session is over, it must be written to a text file with the file name format 2023_12_05_09_56_35_4587.txt . Where, the first part 2023_12_05 denotes the current date, the second part 09_56_35 denotes the time, the third part 4587 is just a random number within the range of 0000 to 9999. There should be one text file for each game session. The text file should record and have attempt number, presented enemies, user input number, WON/LOST status for each attempt, life score and finally the end game statistics.

2. Solution Outline

2.1 Algorithm for the solution

1. Start
2. Import necessary modules random, datetime.
3. Initialize variables to store current date, time, random number, attempt count, life score, final score, user selection, game status, player name, file name.
4. Create a list to to store fighting numbers.
5. Generate a unique file name by combining current date, time, random number.
6. Define a function called ‘prompt()’ to initiate game.
7. Open the created file to record game statistics.
8. Display a welcome message and ask the user to enter the player name and assign it to variable name don.
9. Randomly generate and display the player’s initial life score in range 1-50.
10. Create a while loop to iterate up to maximum 20 attempts to play the game.
11. Generate a list of fighting numbers according to the attempt count.
12. Ask the user to select a number to fight with.
13. If the number selected by the player match with any number on the generated list the program should state whether the player killed the number or else the number killed the player.
14. If the number selected from list is <= life score then that selected value should get add with the life score and proceed to next attempt.
15. If the number selected by the player > life score or number does not match with any number generated in list the program should stop iterating and display the game statistics.
16. The number selected is not in list program should display as “no such enemy”
17. If player type any characters other than integer the program will stop and error message will appear.
18. Record game statistics, including attempts, selected numbers, enemies, and life scores, in the file.
19. If player survived through out all 20 attempts the game status should display as he saved letter kind. Else, should display as he is defeated.
20. Set the final game status based on the outcome.

21. Exit the prompt() function.
22. Create a function named game_statistics(fo) to display and record final game statistics.
23. Display the player name, total attempts, final score, and game status and write the same statistics to the file created.
24. Call the prompt() function to initiate and play the game.
25. Call the game_statistics(fo) function to display and record game statistics.
26. Close the opened file
27. End

2.2 Actual python codes in 20230297.py

```
#Importing modules
import random
from datetime import datetime

#Initializing variables
current_date=0
current_time=0
random_number=0
attempt=0
life_score=0
final_score=0
selection=0
status=()
don=()
filename=()

#Creating a list for fighting numbers
fighting_numbers=[]

#Generating the file name with date,time and random number
current_date = datetime.now().strftime("%Y_%m_%d")
current_time = datetime.now().strftime("%H_%M_%S")
random_number = str(random.randint(0, 9999)).zfill(4)
filename = current_date + "_" + current_time + "_" + random_number + ".txt"

#Simple Function
```

```
def prompt():
    #Declaring global variables
    global attempt
    global final_score
    global don
    global selection
    global status
    global fo



    #Opening the file for recording stats
    fo= open(filename,"w")

    #interface and process
    print("Welcome to destroyer of numbers!!!")
    don=input("player name: ")
    fo.write("Player name: "+don)
    life_score=random.randrange(1,50)
    print(don, "'s initial life score is: ", life_score)

    while attempt<20:
        attempt=attempt+1
        fo.write("\n\nAttempt number: "+str(attempt))
        print("\nAttempt: ",attempt)
        print("Life score is: ",life_score)


        if 1<=attempt<=5:
            fighting_numbers=[random.randrange(15,100) for i in range (5)]
        elif 6<=attempt<=10:
            fighting_numbers=[random.randrange(250,2000) for i in range (5)]
        elif 11<=attempt<=15:
            fighting_numbers=[random.randrange(3000,10000) for i in range (5)]
        elif 16<=attempt<=20:
            fighting_numbers=[random.randrange(20000,100000) for i in range (5)]

        fo.write("\nPresented enimies: "+str(fighting_numbers))
        print("Numbers to fight: ", *fighting_numbers,sep=" ")


        #Error handling for user selection number
        try:
            selection=int(input("Select a number to fight: "))
            fo.write("\nUser input number: "+str(selection))
        except ValueError:
```

```
            print("selection should be an integer")
            break



        if selection in fighting_numbers:
            if selection<=life_score:
                print(don, "killed", selection)
                life_score=life_score+selection
                final_score=life_score
            else:
                print(selection, "killed", don)
                final_score=life_score
                break
        else:
            print("No such enemy")
            final_score=life_score
            status=(don, "was defeated!!!")
            break

        fo.write("\nLife score: "+str(life_score))


    if attempt==20:
        status=(don,"saved letter-kind")
    else:
        status=(don,"was defeated!!!")
    return


#Simple Function
def game_statistics(fo):
    print("\n***Game status***")
    print("Player name:",don)
    print("Total attempts:",attempt)
    print("Final score:",final_score)
    print(*status)

    fo.write("\n\n***Game status***")
    fo.write("\nPlayer name:"+str(don))
    fo.write("\nTotal attempts:"+str(attempt))
    fo.write("\nFinal score:"+str(final_score))
    fo.write("\n"+str(status))
    return
```

```
#Calling the functions
prompt()
game_statistics(fo)

#Closing the file
fo.close()
```

2.3 Screenshots of working program

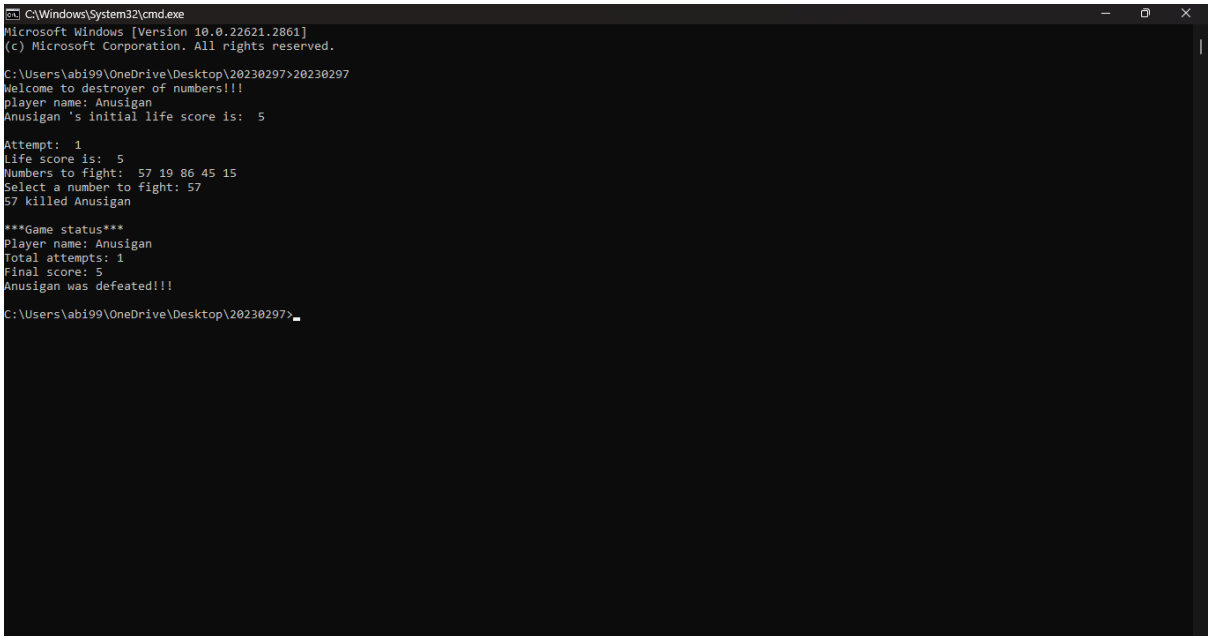

Figure 1-player was defeated

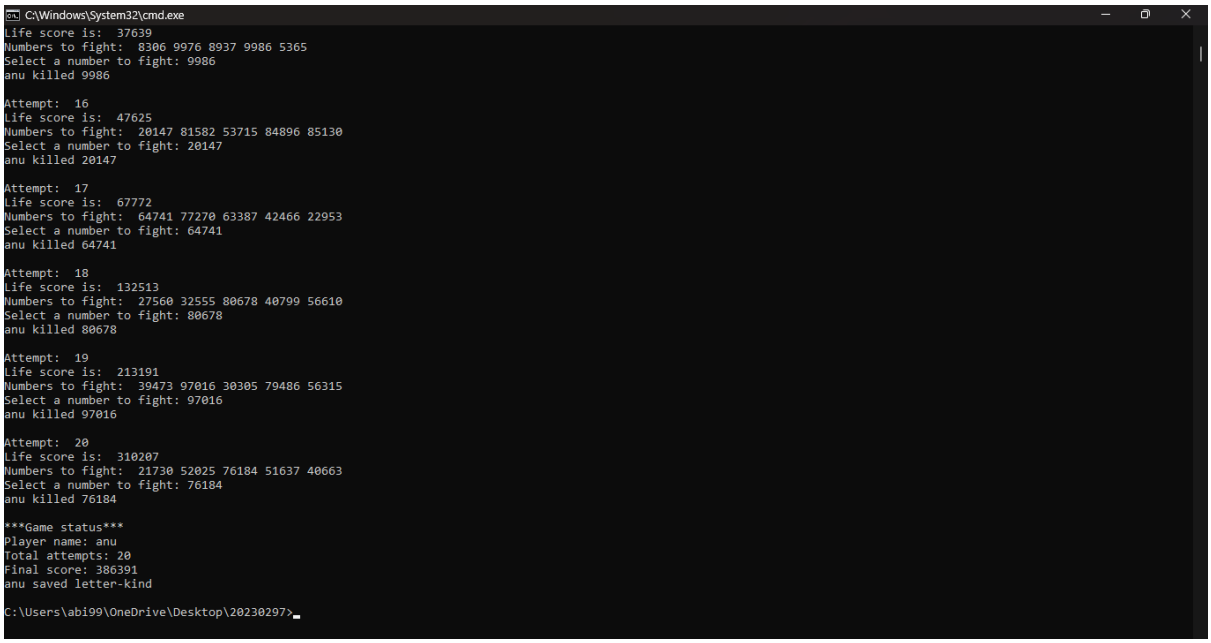

Figure 2-Player saved letter-kind

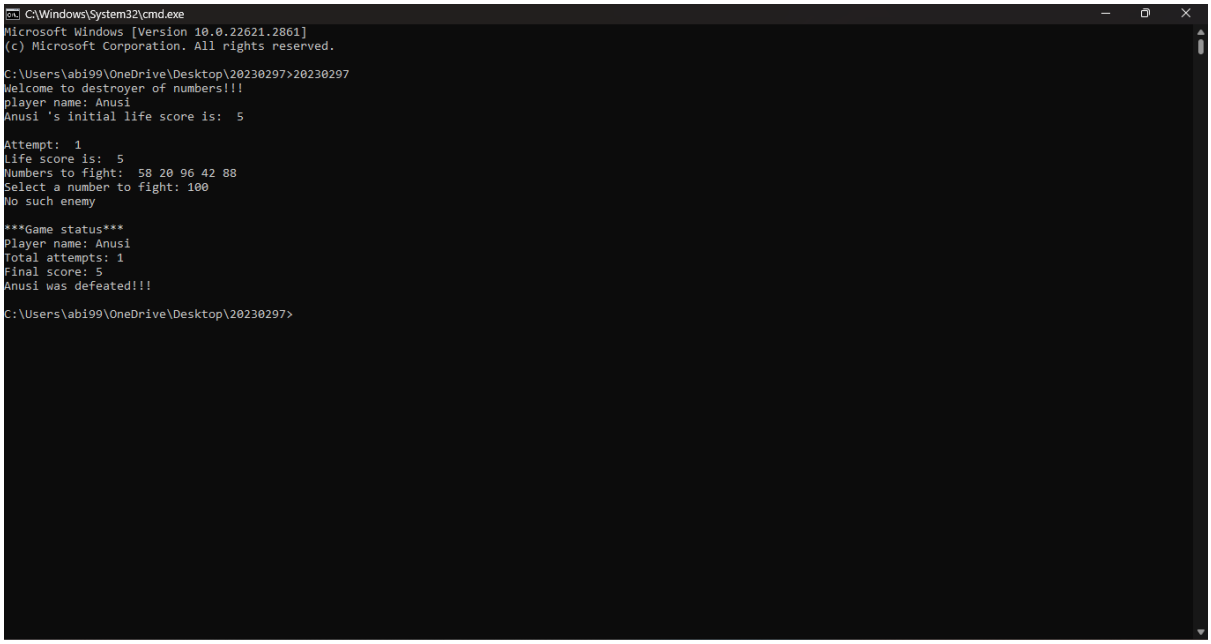

Figure 3-No such enemy

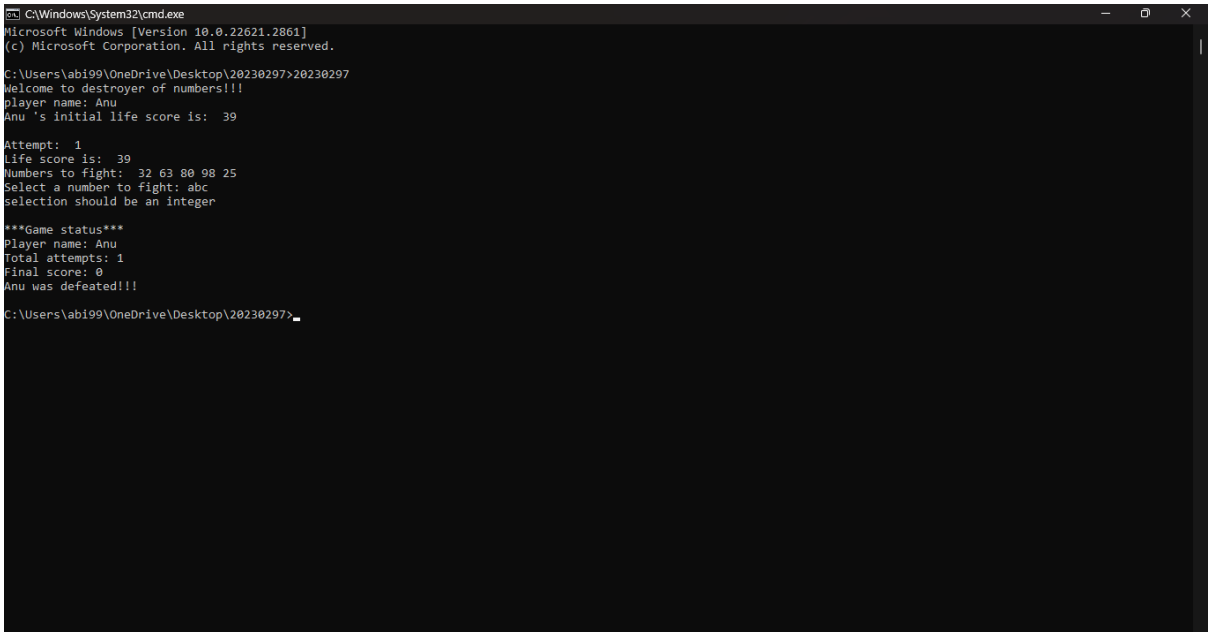

Figure 4-Integer selection required

3. Test Cases

Test case no	Description	Expected Output	Actual Output	PASS / FAIL
1	Player input the name	Initial life score should be generated randomly Attempt 1 should commence asking for a number to fight from 5 numbers listed	Initial life score generated Attempt 1 commenced asking to select a number to fight from list	PASS
2	Player selects a number greater than player's life score	Should display "(selected number) killed (player name)" Game status should be displayed Should display "(player name) was defeated!!!"	Displayed "(selected number) killed (player name)" Game status displayed Displayed "(player name) was defeated!!!"	PASS
3	Player selects a number lower than or equal to player's life score	Should display "(player name) killed (selected number)" Should proceed to next attempt	Displayed "(player name) killed (selected number)" Proceeded to next attempt	PASS
4	Player enters a number which is not presented as selected number to fight	Should display as "no such enemy" Game status should be displayed	Displayed as "no such enemy" Game status displayed	PASS

5	Player enters any character other than integer as numbers selected number to fight	Should display as “selection should be an integer” Game status should be displayed	Displayed as “selection should be an integer” Game status displayed	PASS
6	Player successfully completes all 20 attempts	Game status should be displayed Should display “(player name) saved letter-kind”	Displayed game status Displayed “(player name) saved letter-kind”	PASS
7	Creating text file and record all game statistics	File should be created with current date time and a random number as file name and game statistics should be recorded in it	File created and game statistics are stored in it	PASS
8	One text file for each session	Only one file should be created for each game session	Only one file created for each session	PASS
9	Checking the program in Windows command console	Should run successfully	Program successfully ran in windows command console	PASS
10	Checking the program with IDLE	Should run successfully	Program successfully ran in IDLE	PASS

Table 1-Test Cases

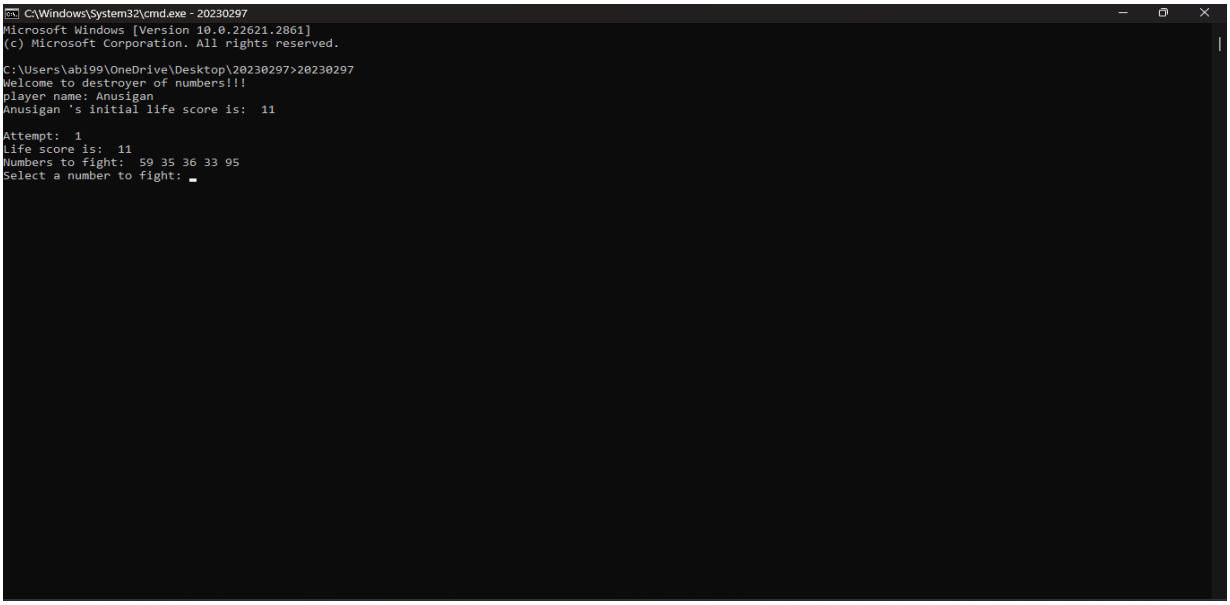

Figure 5-Screenshot proof for test case 1

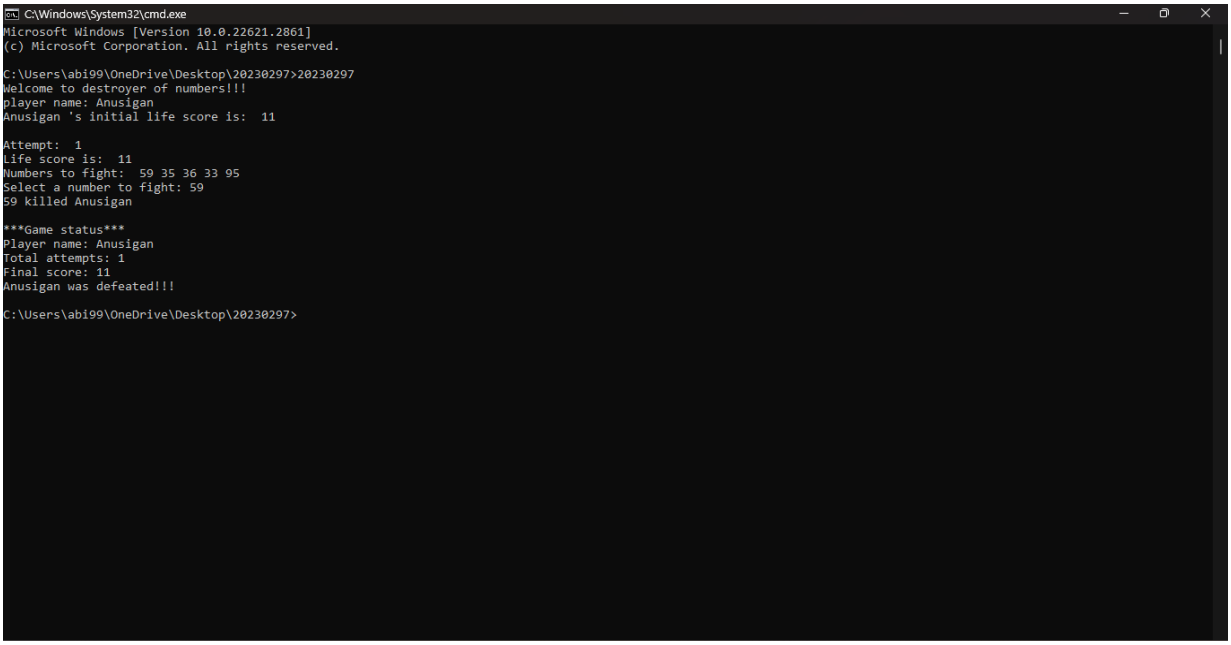

Figure 6-Screenshot proof for test case 2

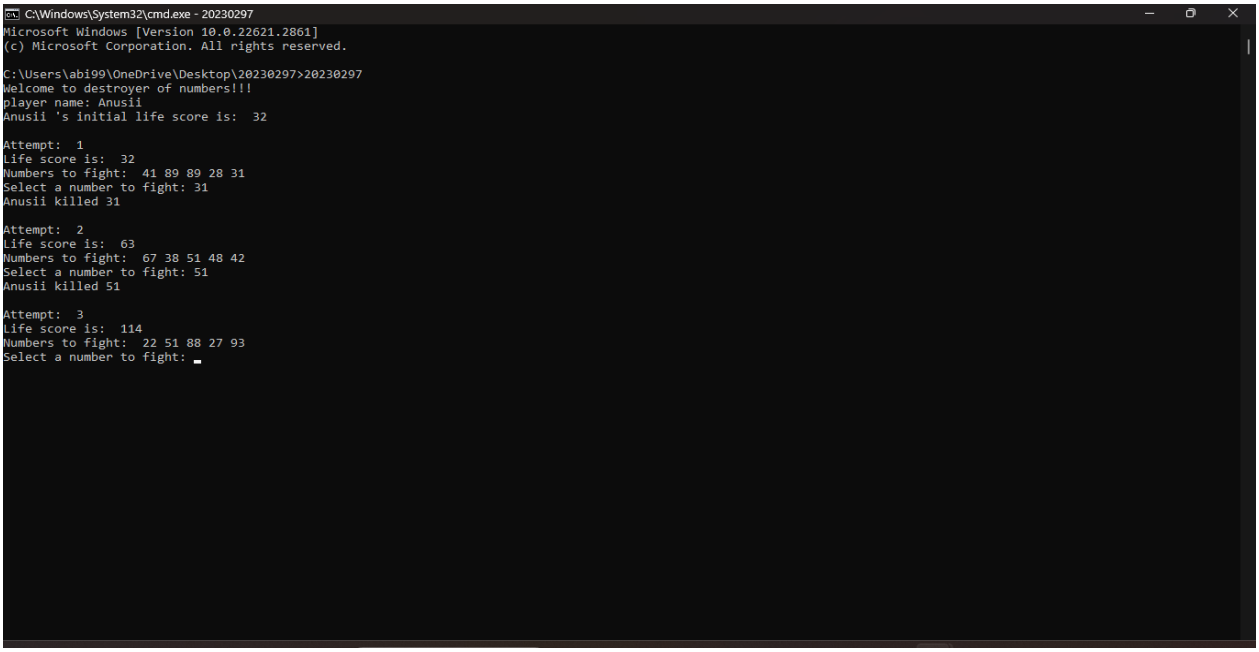

Figure 7-Screenshot proof for test case 3

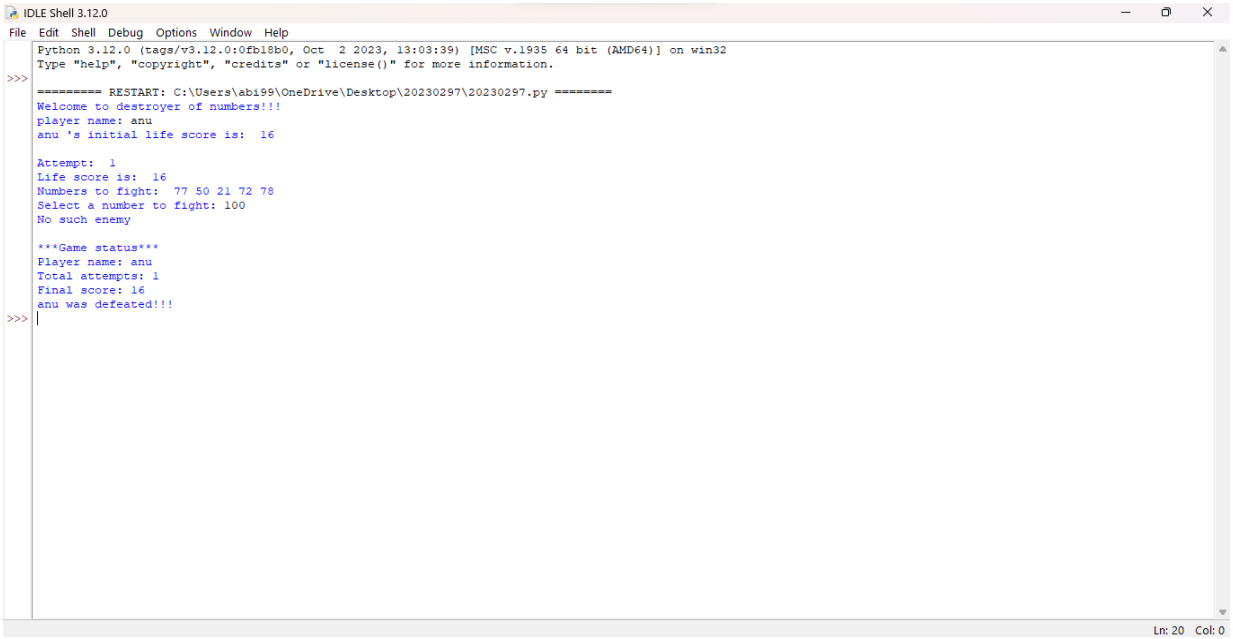

Figure 8-Screenshot proof for test case 4

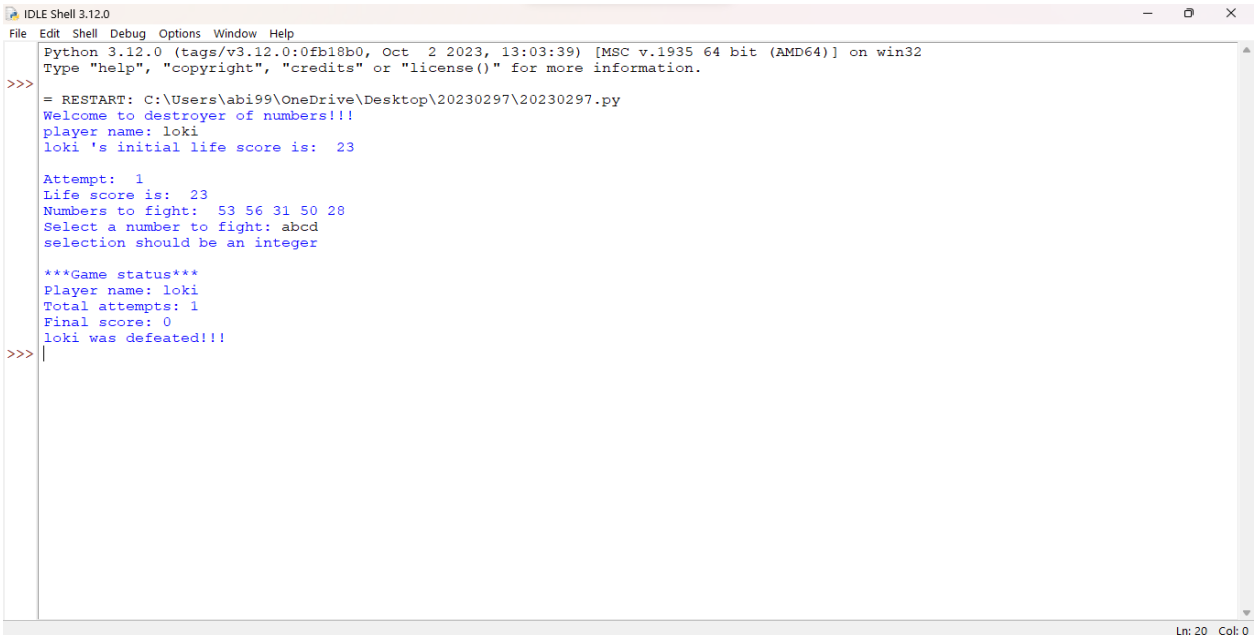

Figure 9-Screenshot proof for test case 5

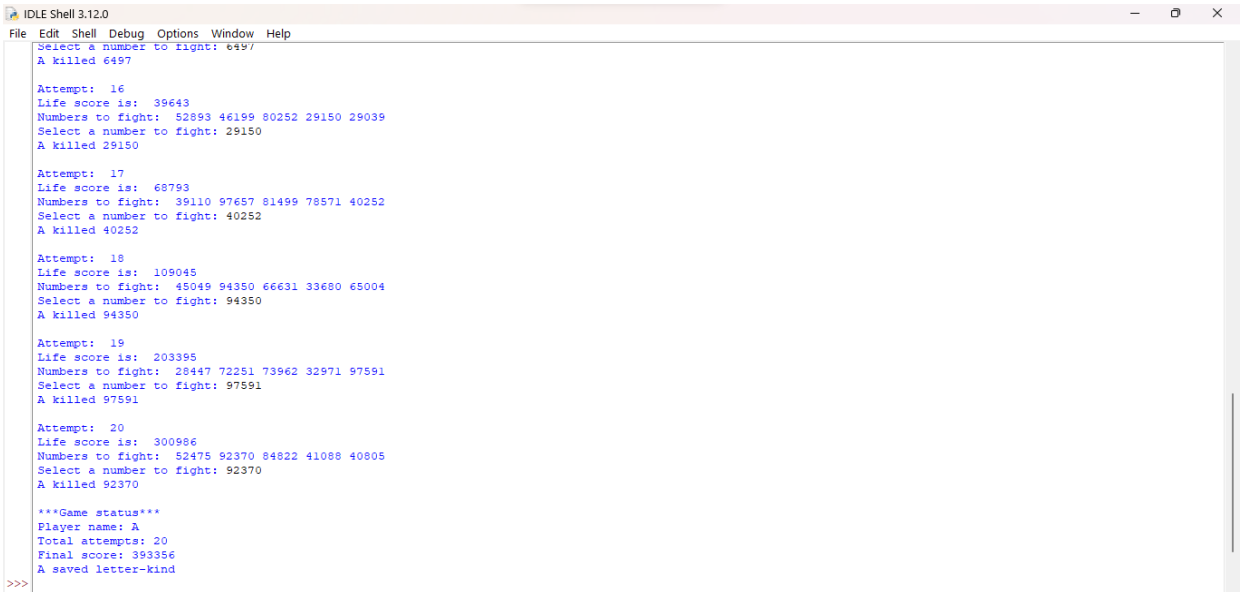

Figure 10-Screenshot proof for test case 6

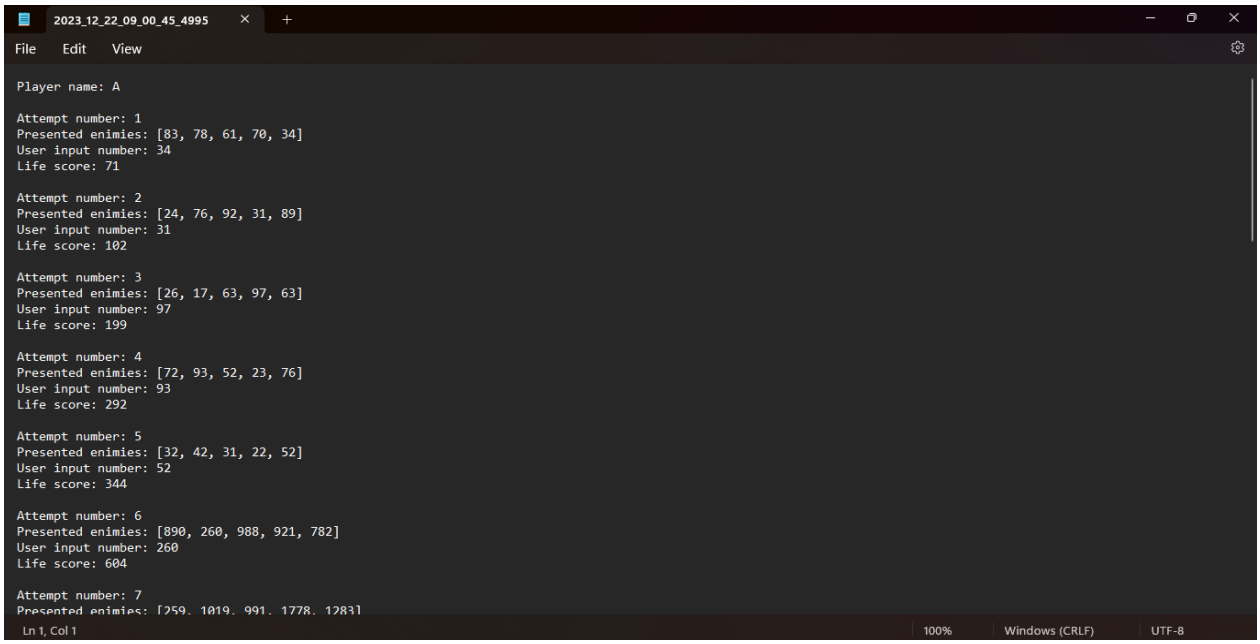

Figure 11-Screenshot proof for test case 7

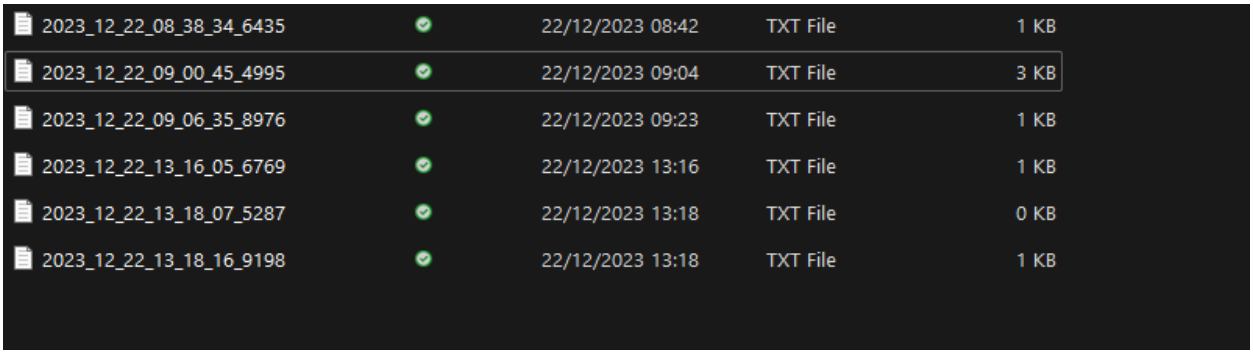

Figure 12-Screenshot proof for test case 8

Figure 13-Screenshot proof for test case 9

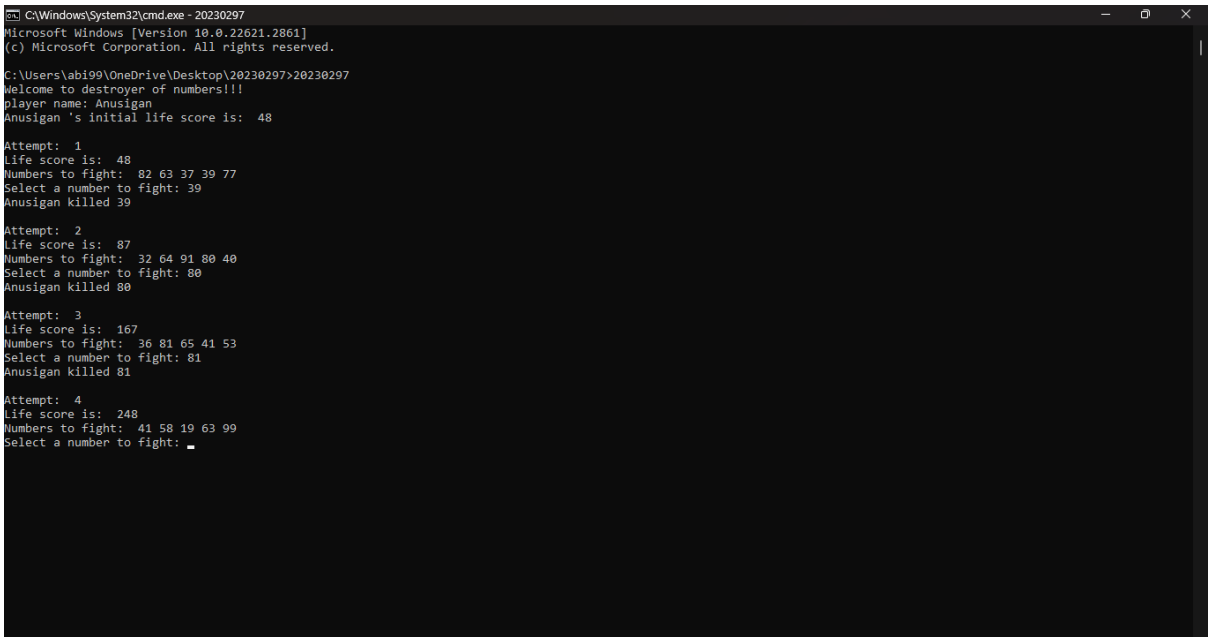

Figure 14-Screenshot proof for test case 10

4. Conclusion

In conclusion, the Python coursework project "DON: Destroyer of Numbers" has been effectively executed, showcasing the application of fundamental programming concepts in a game development context. The coursework has provided valuable hands-on experience in developing interactive applications using Python, refining problem-solving skills and reinforcing a structured approach to programming. Overall, the successful completion of "DON: Destroyer of Numbers" marks a milestone in the application of Python programming skills, combining creativity with technical proficiency in the development of an engaging text-based game.