
INFORMATICS INSTITUTE OF TECHNOLOGY

# Foundation Certificate in Higher Education

**Module** – DOC334 Computer Programming

**Course Leader** – Mr. Nishan Saliya Harankahawa

**Type of Assignment** – Individual Course Work

**Date of Submission** – 29th of November 2024

**Student ID** – 20232674

**Student Name** – M.P.M.N.Gunarathna

**Level** – 2nd Semester Foundation(Colombo)

# Abstract

This project introduces MathBro, an interactive Python-based math test app that aims to improve users' math abilities by providing a fun, adaptable learning environment. Users can choose from four different difficulty levels in the application ( demo, easy, medium, and hard). Depending on the mode they choose, the application will generate arithmetic questions at random. Every session is recorded, along with the user's answers, accuracy, and performance indicators like scores.

# Acknowledgement

To begin, I would like to sincerely thank Mr. Nishan Saliya for seeing my potential and giving me the chance to work on this project. In order to successfully complete this work, his unwavering support, encouragement, and invaluable guidance have been essential. This accomplishment would not have been possible without his guidance, for which I am sincerely thankful.

Additionally, I want to express my sincere gratitude to my parents for their unwavering sacrifices and hard work in guaranteeing a better future for me. Their constant encouragement and support have been my biggest source of inspiration during this journey.

Finally, I want to sincerely thank everyone for their ongoing support and meticulous attention to detail, which have been crucial to the successful completion of this report.

Thank You.

# Table of Contents

## Contents

# List of figures

# List of Tables

# 1. Introduction

This course is dedicated to the development of MathBro, a fun and interactive Python application aimed at enhancing users' math skills. The project offers dynamic quizzes at different difficulty levels in an effort to give users of all ages an enjoyable learning experience. MathBro leverages the powerful features of Python to ensure adaptability, ease of use, and efficient performance monitoring.

Math can be much less intimidating or overwhelming and more accessible as the need for educational resources that blend enjoyment and usefulness is continuously growing. In the settings, users have the ability to select different difficulty levels: Demo, Easy, Medium, and Hard. Each level was fitted to a different skill and difficulty levels. MathBro allows one practice by enabling features such as preserving the session data, performance analysis, and random question generation for continuous improvements in practice over time.

This project combines some of the most important concepts in programming, including control structures, file handling, and modular design, using Python in a way that solves real-world challenges. Also, the application displays the results of sessions both in text and HTML formats, showing proficiency in handling multiple output formats for user convenience.

In this project, aim to be able to highlight how programming could provide new, effective and also put into practice theoretic material that I learned during the course for a practical and realistic solution.

# 2. Algorithm

1. Start

2. Initialize the variables

3. Mode Selection

   If no argument is given, set mode to demo.
   If the argument is **-e** , switch to **Easy mode.**
   If the argument is **-m** , switch to **Medium mode.**
   If the argument is **-h** , switch to **Hard mode**.
   If the argument is not valid , terminate the program with an error message

4. Set up question

   **Demo Mode:**

   Number of questions: 3
   Number range: 1 to 5
   Operators: “+”

   **Easy Mode:**

   Number of questions: 5
   Number range: 1 to 10
   Operators: “ +” , “ – ”

   **Medium Mode:**

   Number of questions: 10
   Number range: 1 to 10
   Operators: “ +” , “ – ”

   **Hard Mode:**

   Number of questions: 10
   Number range: 1 to 20
   Operators: “ +” , “-” , “ * ”

5. Question Generation

   Randomly pick two numbers and an operator.
   Display questions and get user input.
   Validate input and evaluate the answer.
   Track correct/wrong answers and update session.
   Display session results.
   Ask if the user wants another session. Repeat or exit

6. Saving Session Data

   Save session results in a **text file**
   Save session results in an **HTML file**

7. Main Function

   Call mode_selector ( ) to get the mode.
   Call sys_configurations ( ) to configure the system.
   To start the session and manage user input, question creation, and session tracking, call question_generator ( ).

8. End.

## 3. Python Code

```
import sys
import random
import datetime
import webbrowser
import os

#Initialize global variable

global session_id
session_id = 0

def mode_selector():
    mode = ""

    if len(sys.argv) < 2:
        mode = "demo"
    else:
        #Map the arguments to respective modes
        if sys.argv[1] == "-e":
            mode = "easy"
        elif sys.argv[1] == "-m":
            mode = "medium"
        elif sys.argv[1] == "-h":
            mode = "hard"
        else:
            print("Invalid mode selected.")
```

```
            sys.exit()

    return mode

def sys_configurations(mode):

    arithmatics = ["+"]
    start_range = 0
    end_range = 0
    number_of_questions = 0

    match mode:
        case "demo":
            start_range = 0
            end_range = 5
            number_of_questions = 3
        case "easy":
            start_range = 0
            end_range = 10
            number_of_questions = 5
            arithmatics.append("-")
        case "medium":
            start_range = 0
            end_range = 10
            number_of_questions = 10
            arithmatics.append("-")
        case "hard":
            start_range = 0
            end_range = 20
```

```
        number_of_questions = 10
        arithmatics.append("-")
        arithmatics.append("*")

    return arithmatics, start_range, end_range, number_of_questions,mode

def quection_generator(arithmatics, start_range, end_range, number_of_questions, mode):

    all_sessions = []
    global session_id

    while True:
        session_id = session_id + 1
        session = {"sessionId": session_id, "sessionData": [], "mode": mode,"Correct": 0, "Wrong": 0}
        correct = 0
        wrong = 0


        print(f"\n\nSession {session_id} Mathbro.\n\n")


        for i in range(number_of_questions):

            num1 = random.randint(start_range, end_range)
            num2 = random.randint(start_range, end_range)
            operator = random.choice(arithmatics)

            quection = f" | {i+1}) {num1} {operator} {num2} = ? "
            answer = answer_generator(num1, num2, operator)
```

```
    while True:
        try:
            user_answer = int(input(quection))
            break
        except ValueError:
            print("Invalid input! Please enter a numeric value.")

    if user_answer == (answer):
        result = "(√) You're answer is Correct! \n"
        mark = "√"
        txt_result = f"{mark} {quection} ={user_answer}"
        correct = correct + 1

    else:
        result = f"(X) You're answer is Wrong! correct answer is {str(answer)} \n"
        mark = "X"
        txt_result = f"{mark} {quection} = {user_answer}  Correct answer is {answer}"
        wrong = wrong + 1

    print(result)
    session["sessionData"].append([txt_result])
#Update session status
session["mode"] = mode
session["Correct"] = correct
session["Wrong"] = wrong

all_sessions.append(session)
print(f"Session {session_id} completed. Correct: {correct}, Wrong: {wrong}")
```

```
        another_session = input("Do you want to play another session? (y/n)")

        if another_session != "y":
            print("Thanks for playing!")
            break

    save_session_txt(all_sessions)

# Function to save session data to a text file
def save_session_txt(all_sessions):
    file_name = f"{datetime.datetime.now().strftime('%Y%m%d_%H%M')}_{random.randint(100, 999)}.txt"

    with open(file_name, "w", encoding="utf-8") as file:
        file.write("MathBro Session\n")
        file.write("--------------------\n\n")

        file.write("Date : " + datetime.datetime.now().strftime("%Y-%m-%d") + "\n")
        file.write("Time : " + datetime.datetime.now().strftime("%H:%M") + "\n\n")

        # Write session details
        for session in all_sessions:

            file.write(f"Session ID: {session['sessionId']}\n")
            file.write("Results Sheet\n\n")

            for data in session['sessionData']:
```

```
            file.write(data[0] + "\n")

        file.write(f"\n\nMode: {session['mode']}\n")
        file.write(f"Total Questions: {len(session['sessionData'])}\n")
        file.write(f"Correct: {session['Correct']}\n")
        file.write(f"Wrong: {session['Wrong']}\n")
        file.write(f"Score: {session['Correct']*100 / len(session['sessionData']):.2f}\n")
        file.write("\n\n")

    print(f"Session saved as {file_name}")

save_html_body(all_sessions)

# Function to save session data to an HTML file
def save_html_body(all_sessions):
    file_name = f"{datetime.datetime.now().strftime('%Y%m%d_%H%M')}_{random.randint(100, 999)}.html"

    with open(file_name, "w", encoding="utf-8") as file:
        file.write("<html>\n")
        file.write("<head>\n")
        file.write("<title>MathBro Session</title>\n")
        file.write("</head>\n")
        file.write("<body>\n")

        file.write("<h1>MathBro Session</h1>\n")
        file.write("<hr>\n")

        file.write("<h2>Results</h2>\n")
```

```
for session in all_sessions:
    file.write(f"<h3>Session ID: {session['sessionId']}</h3>\n")

    file.write("<h4>Results Sheet</h4>\n")
    file.write("<hr>\n")

    file.write(f"<p><b>Date : {datetime.datetime.now().strftime("%Y-%m-%d")}</b></p>\n")
    file.write(f"<p><b>Time : {datetime.datetime.now().strftime('%H:%M')}</b></p>\n")



    file.write("<table border='1'>\n")
    file.write("<tr><th>Results</th></tr>\n")
    for data in session['sessionData']:
        file.write(f"<tr><td>{data[0]}</td></tr>\n")

    file.write("</table>\n")

    file.write(f"<p>Mode: {session['mode']}</p>\n")
    file.write(f"<p>Total Questions: {len(session['sessionData'])}</p>\n")
    file.write(f"<p>Correct: {session['Correct']}</p>\n")
    file.write(f"<p>Wrong: {session['Wrong']}</p>\n")
    file.write(f"<p>Score: {session['Correct']*100 / len(session['sessionData']):.2f}</p>\n")
    file.write("<hr>\n")

file.write("</body>\n")
file.write("</html>\n")

file.close()
```

```
        # Ask if the user wants to open the file in a browser
        openHtmlDoc = input("Do you want to open the session in browser? (y/n)")
        if openHtmlDoc == "y":
            webbrowser.open(f"file://{os.path.realpath(file_name)}")



        print(f"Session saved as {file_name}")

# Function to calculate the answer for a question
def answer_generator(num1, num2, operator):
    return eval(f"{num1}{operator}{num2}")

# Main function to start the program
def main():
    mode = mode_selector()
    arithmatics, start_range, end_range, number_of_questions, mode = sys_configurations(mode)
    quection_generator(arithmatics, start_range, end_range, number_of_questions, mode)

# Entry point of the program

if __name__ == "__main__":
    main()
```

# 4. Explain Packages

## 4.1 Import os

- The OS module in Python provides functions for interacting with the operating system.
- **Usage in the Program:**

Resolves the absolute path of the generated HTML file so it can be opened in the browser

## 4.2 Import webbrowser

- In Python, **webbrowser module** is a convenient web browser controller. It provides a high-level interface that allows displaying Web-based documents to users
- **Usage in the Program:**

Opens the generated HTML file in the user's default browser for viewing session results.

## 4.3 Import datetime

- The datetime module supplies classes for manipulating dates and times. It provides a combination of the date and time modules for more complex operations involving both dates and times.

- **Usage in the Program**:

Formats the current date and time for file naming and session details.

## 4.4 Import sys

- The sys module used to handle command-line arguments and system-specific functions.

- **Usage in the Program:**

  sys.argv is used to read command-line arguments for selecting the difficulty mode (demo, -e, -m, -h).

## 4.5 Import random

- This module can be used to perform random actions such as generating random numbers, printing random a value for a list or string, etc.

- **Usage in the program:**

random.randint(start_range, end_range): Generates random numbers for math problems.

random.choice(arithmatics): Selects a random operator (+, -, *) from a list.

random.randint(100, 999): Generates a random number to create unique filenames for session data.

# 5. Running Program

## 5.1 Demo mode

### 5.1.1 Input Correct answers only

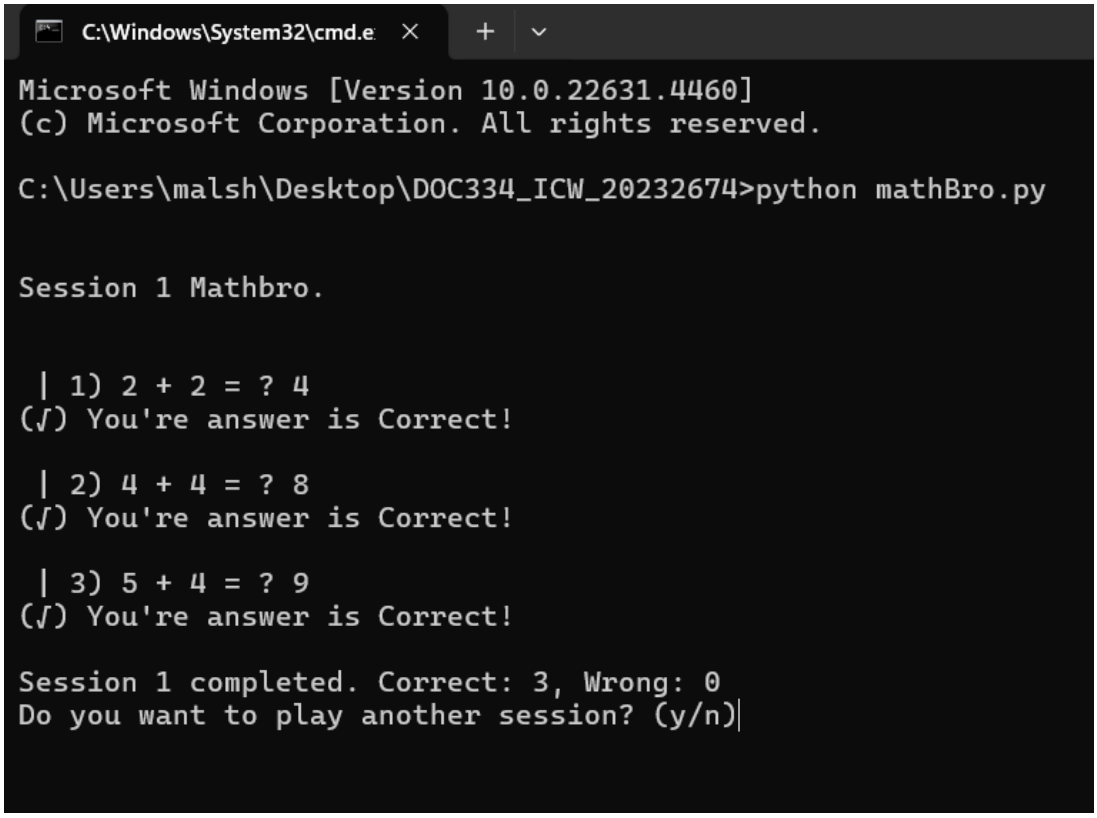


*Figure 1. Correct input - Demo*

### 5.1.2 Input Correct answers , Wrong answers

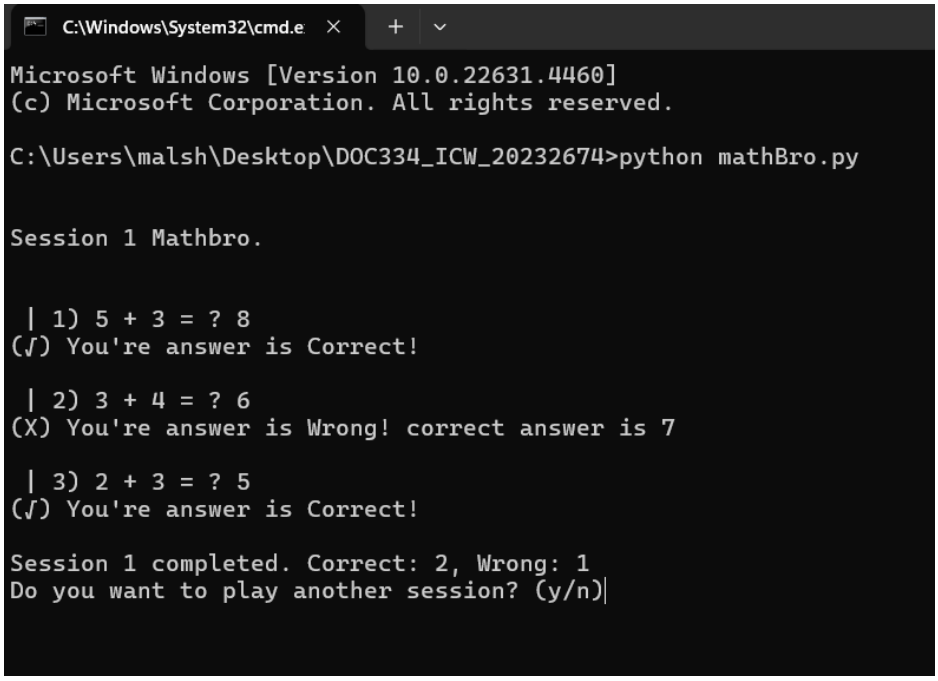


*Figure 2.correct and wrong input -Demo*

### 5.1.3 Input String

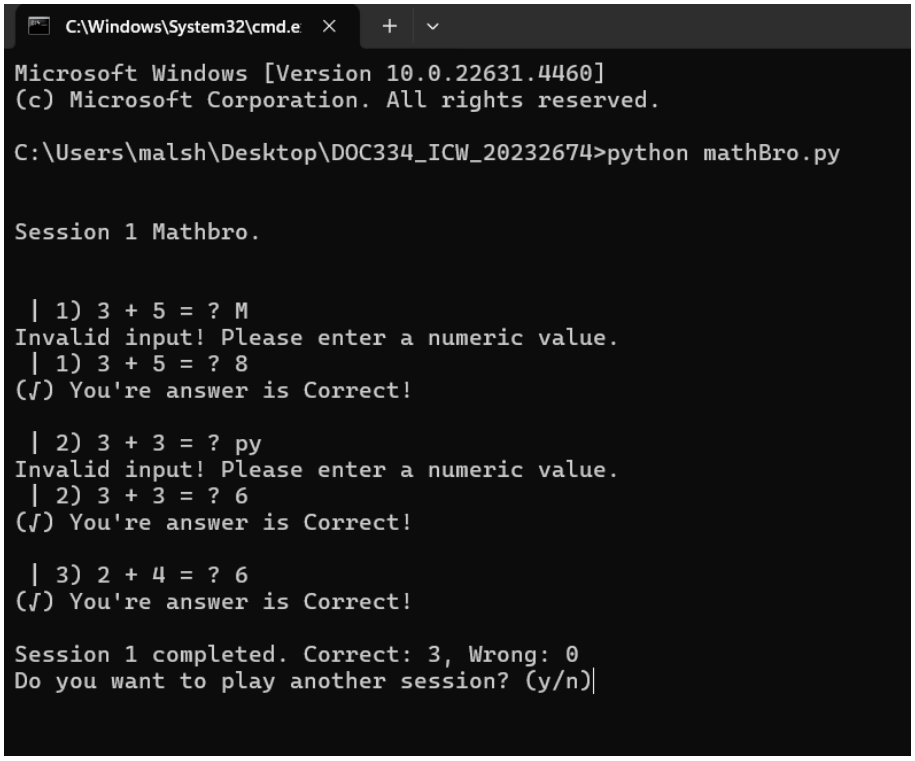


*Figure 3.Input string - Demo*

### 5.1.4 Play another session

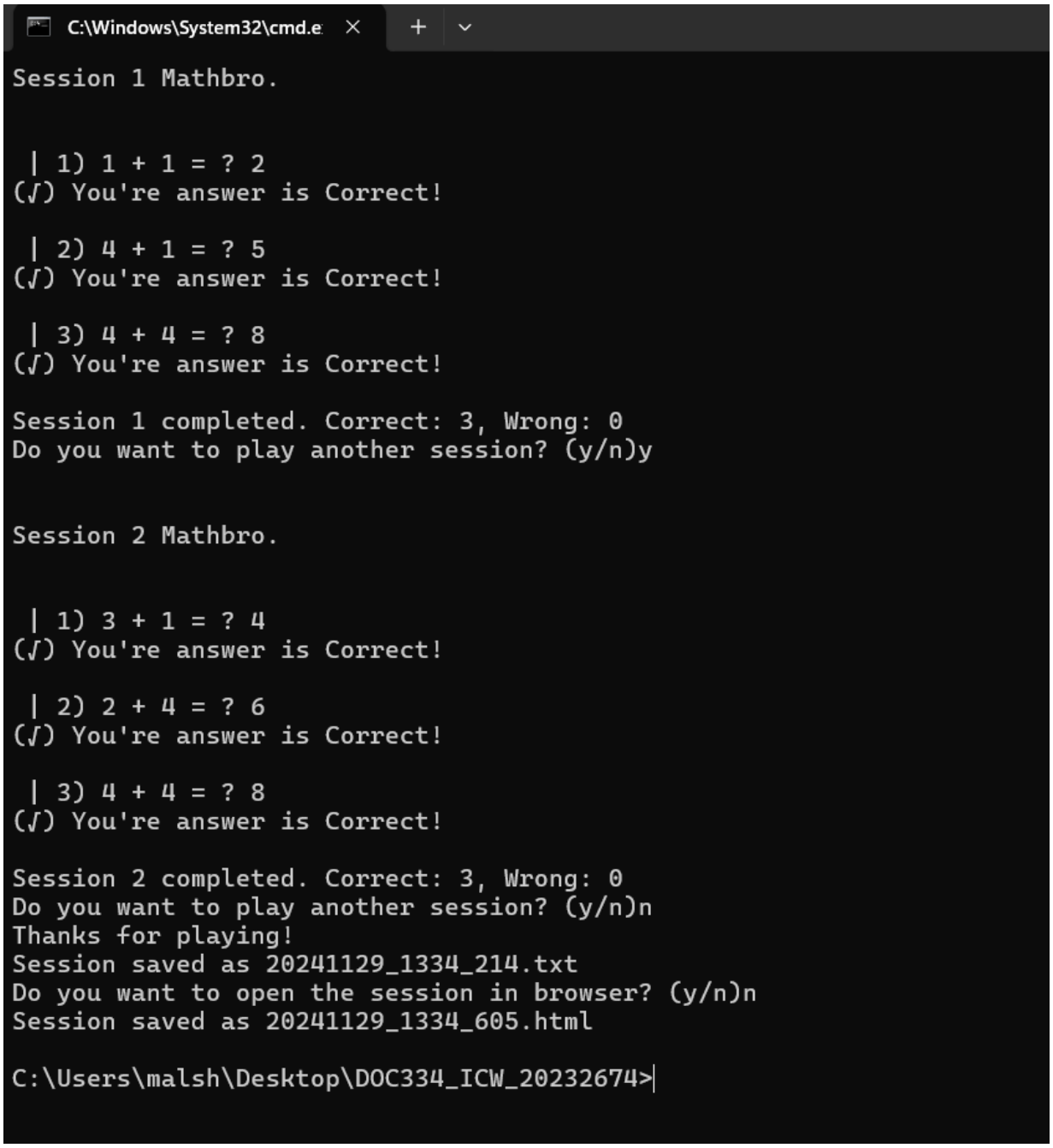


*Figure 4.Play another session - Demo*

## 5.2 Easy Mode

### 5.2.1 Input Correct answers only

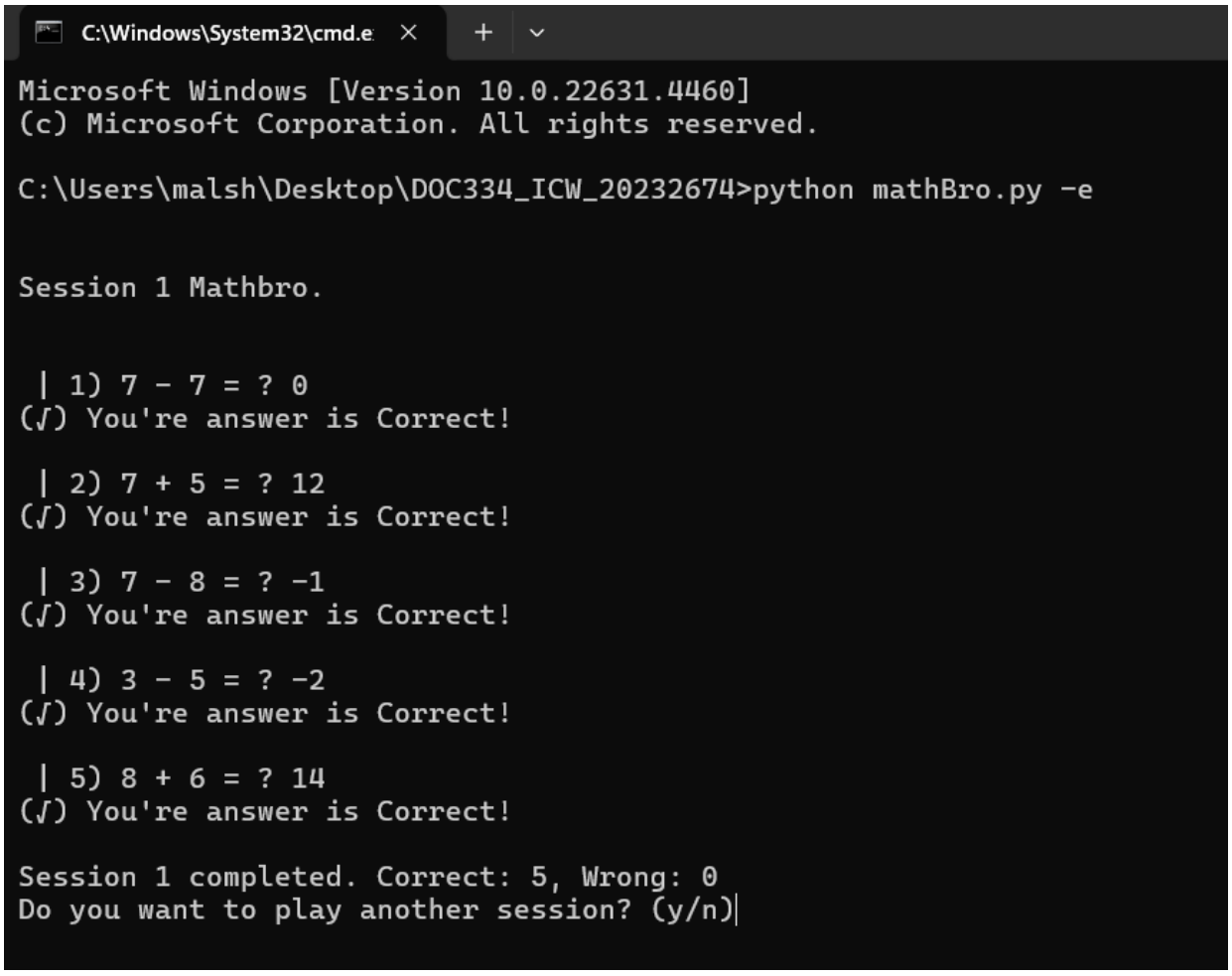


*Figure 5.Correct input -Easy*

### 5.2.2 Input Correct answers , Wrong answers

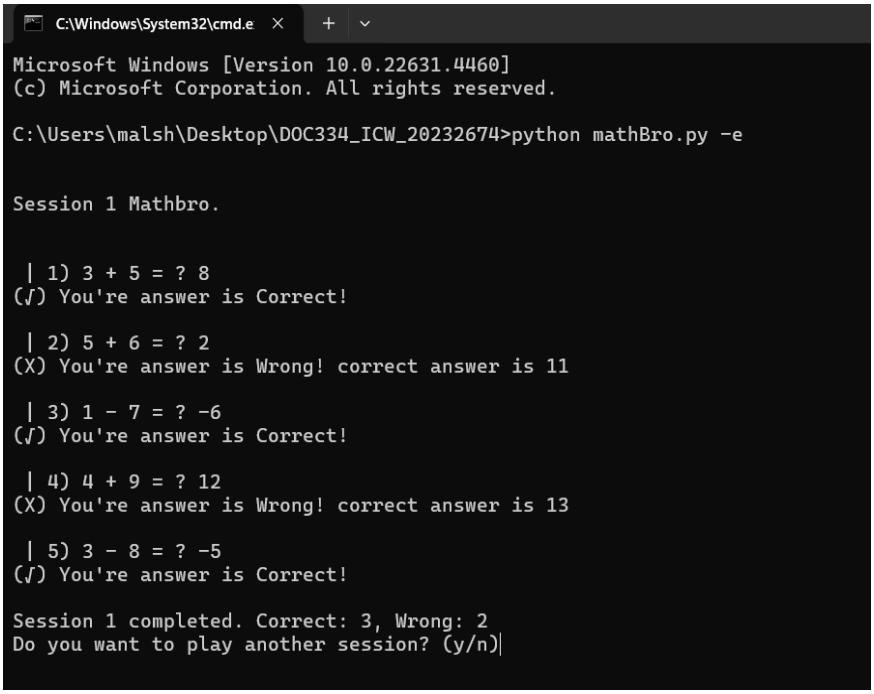


*Figure 6.Correct input & wrong input - Easy*

### 5.2.3 Input string

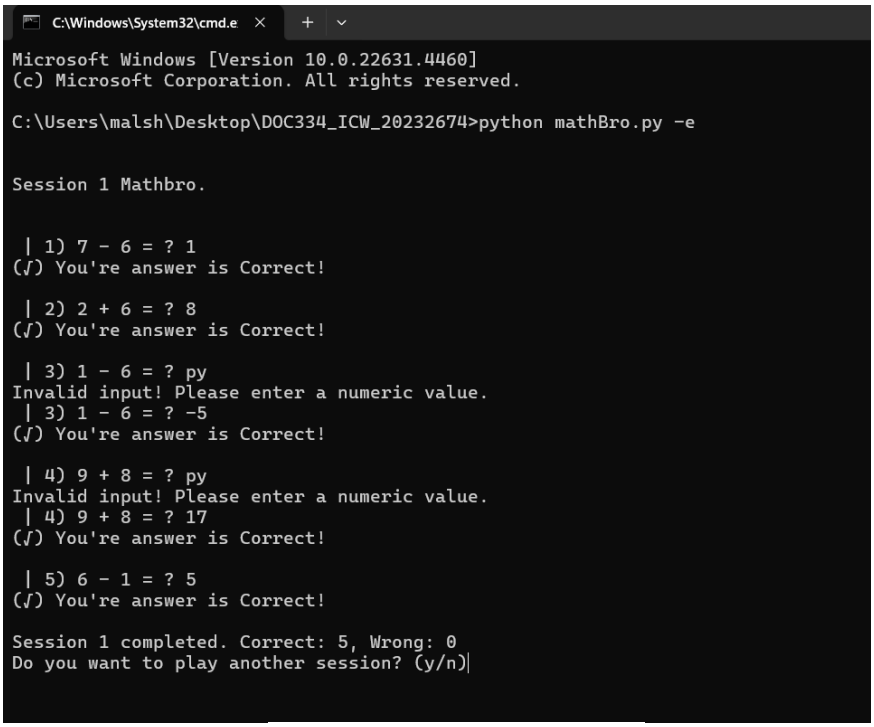


*Figure 7.Input String - Easy*

### 5.2.4 Play another session

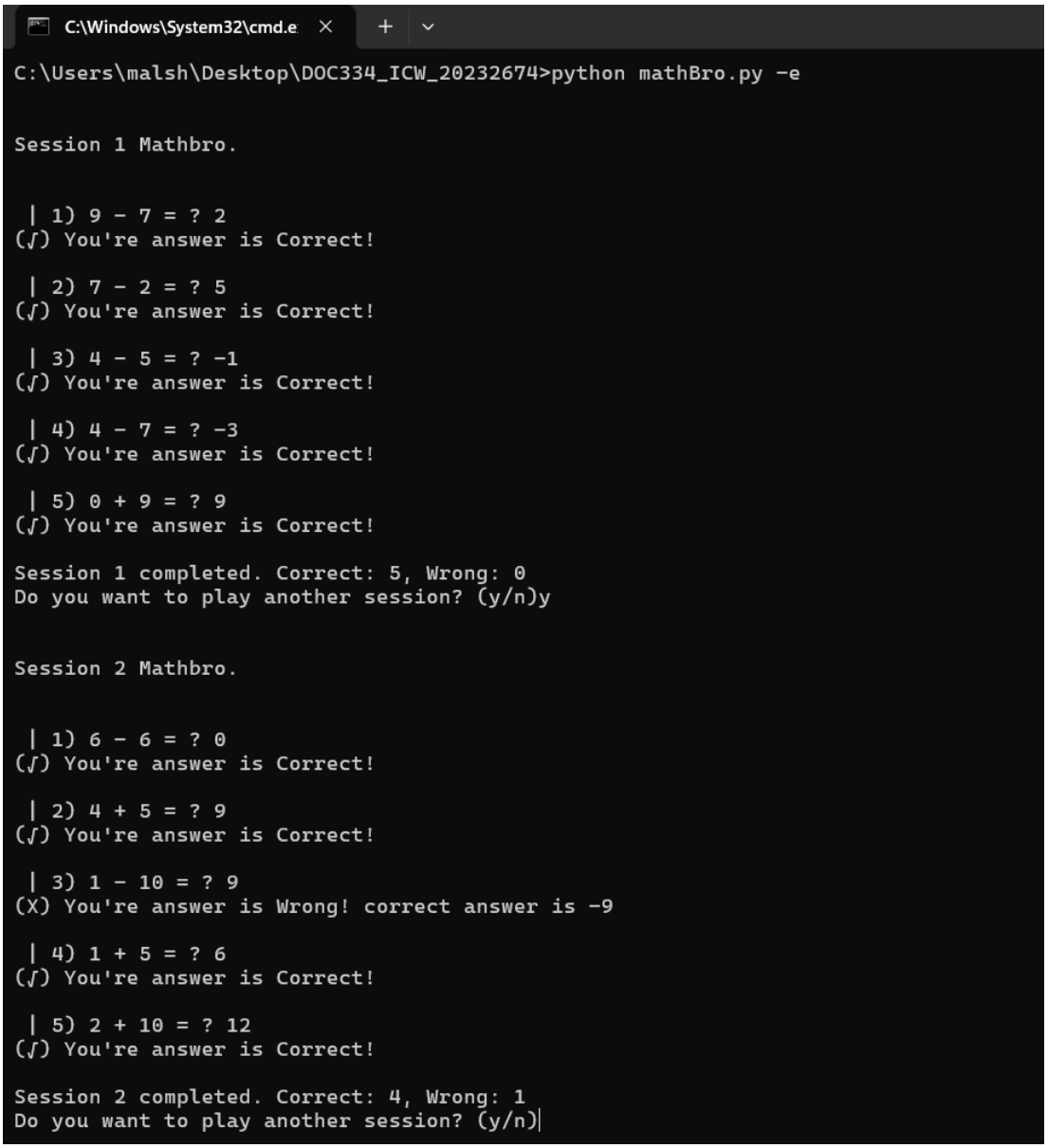


*Figure 8.play another session -Easy*

## 5.3 Medium Mode

### 5.3.1 Input Correct answers only

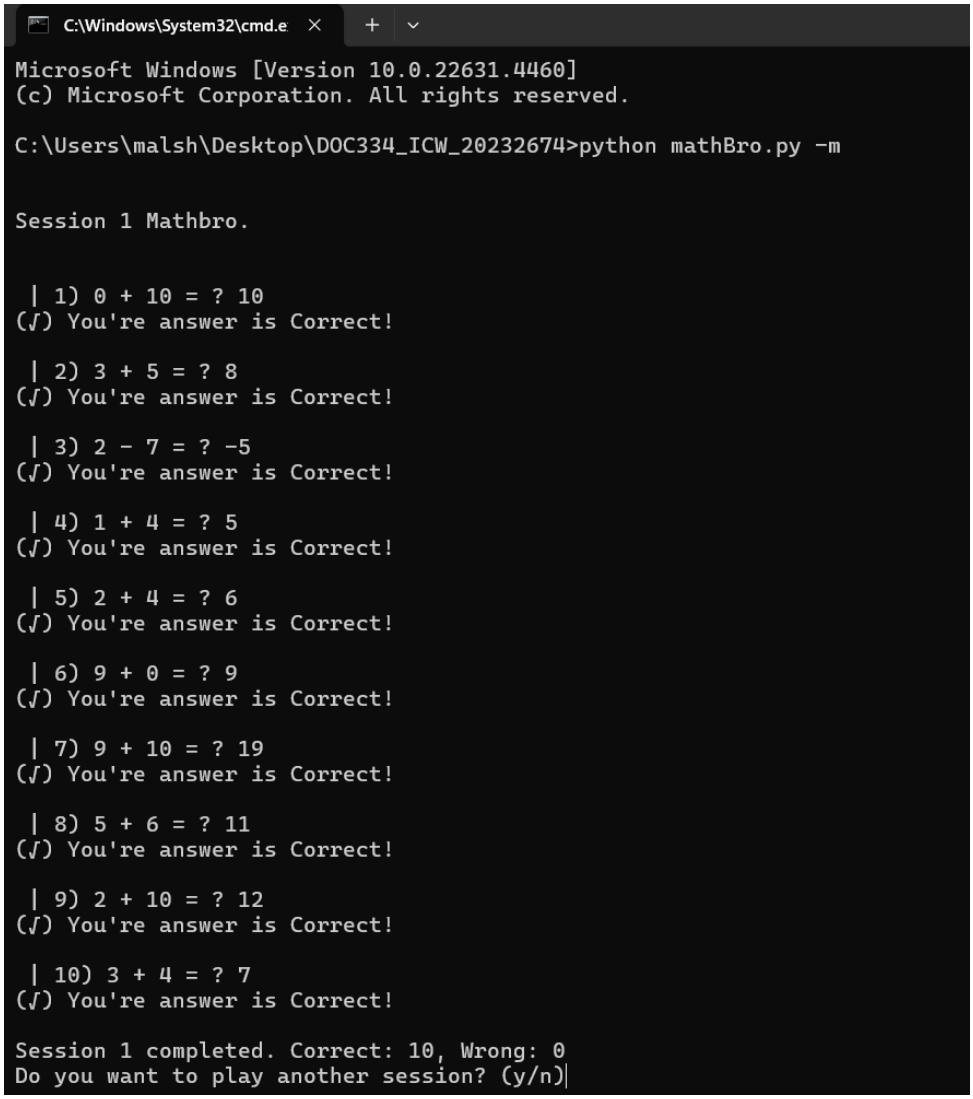


*Figure 9.Correct input -Medium*

### 5.3.2 Input Correct answers , Wrong answers

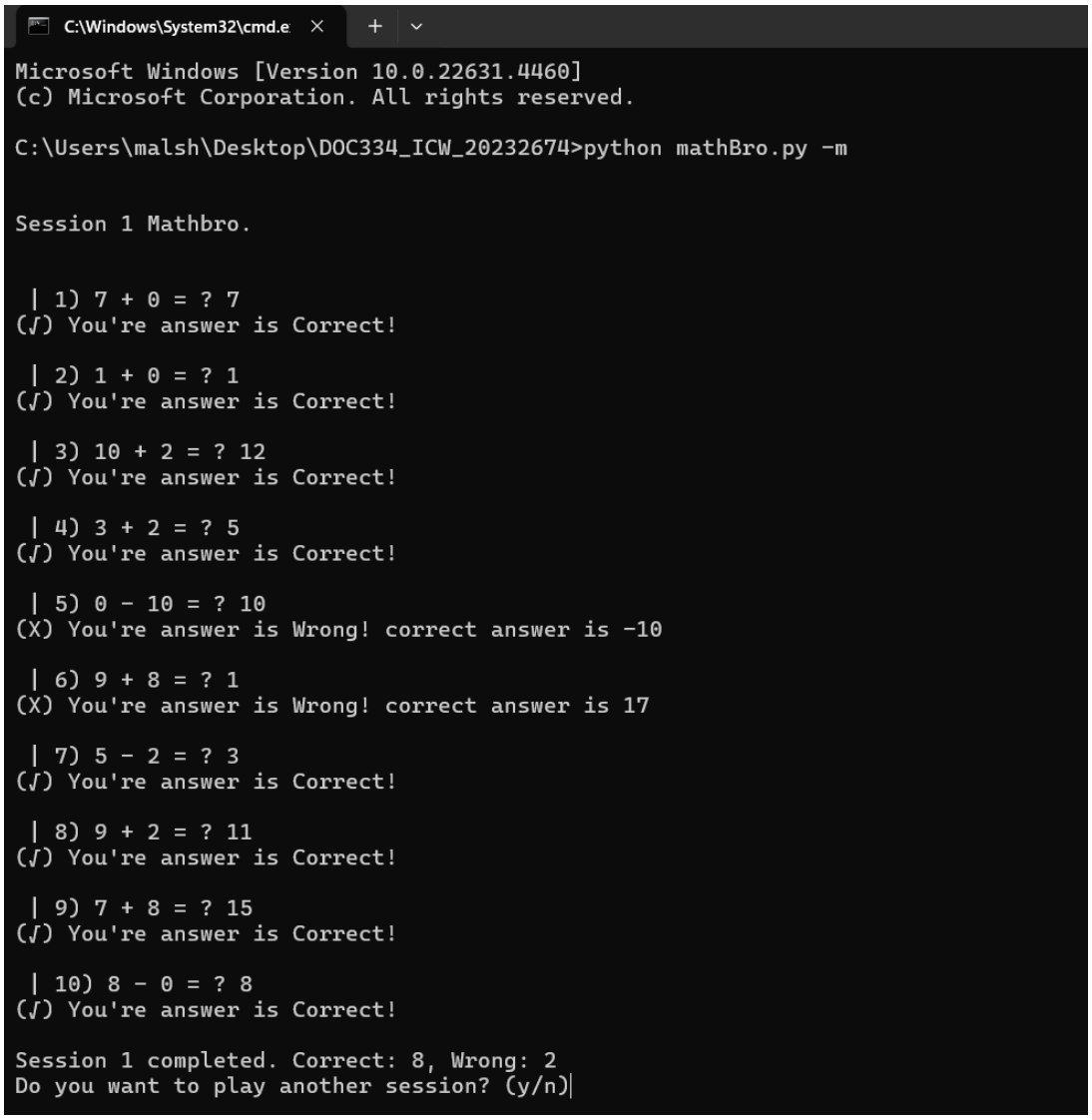


*Figure 10.Correct input & wrong input - medium*

### 5.3.3 Input String

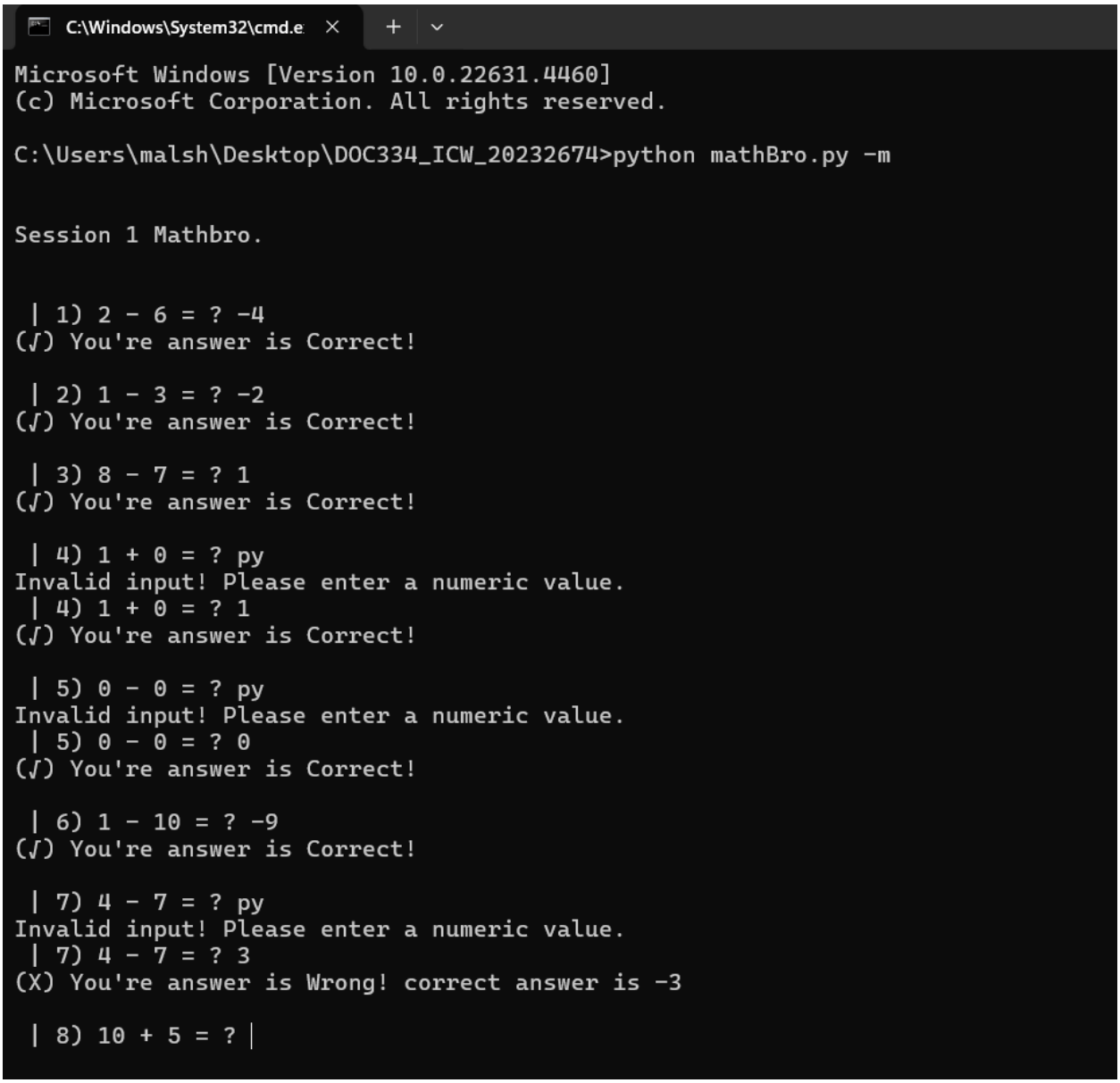


*Figure 11.Input string - medium*

### 5.3.4 Play another session

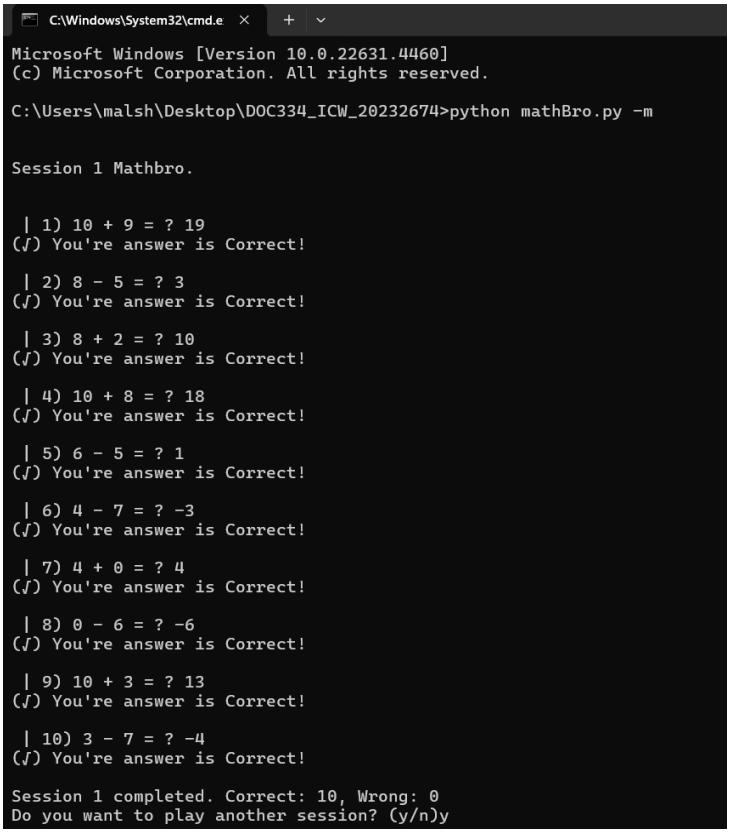


*Figure 12.play another session1 - medium*

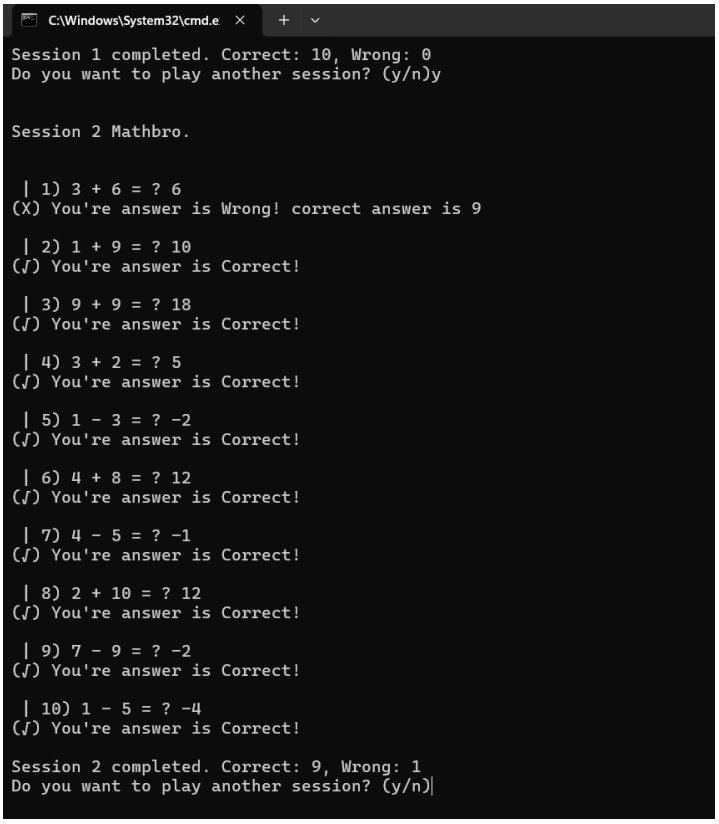


*Figure 13.play another session2 - medium*

## 5.4 Hard Mode

### 5.4.1 Input Correct answers only

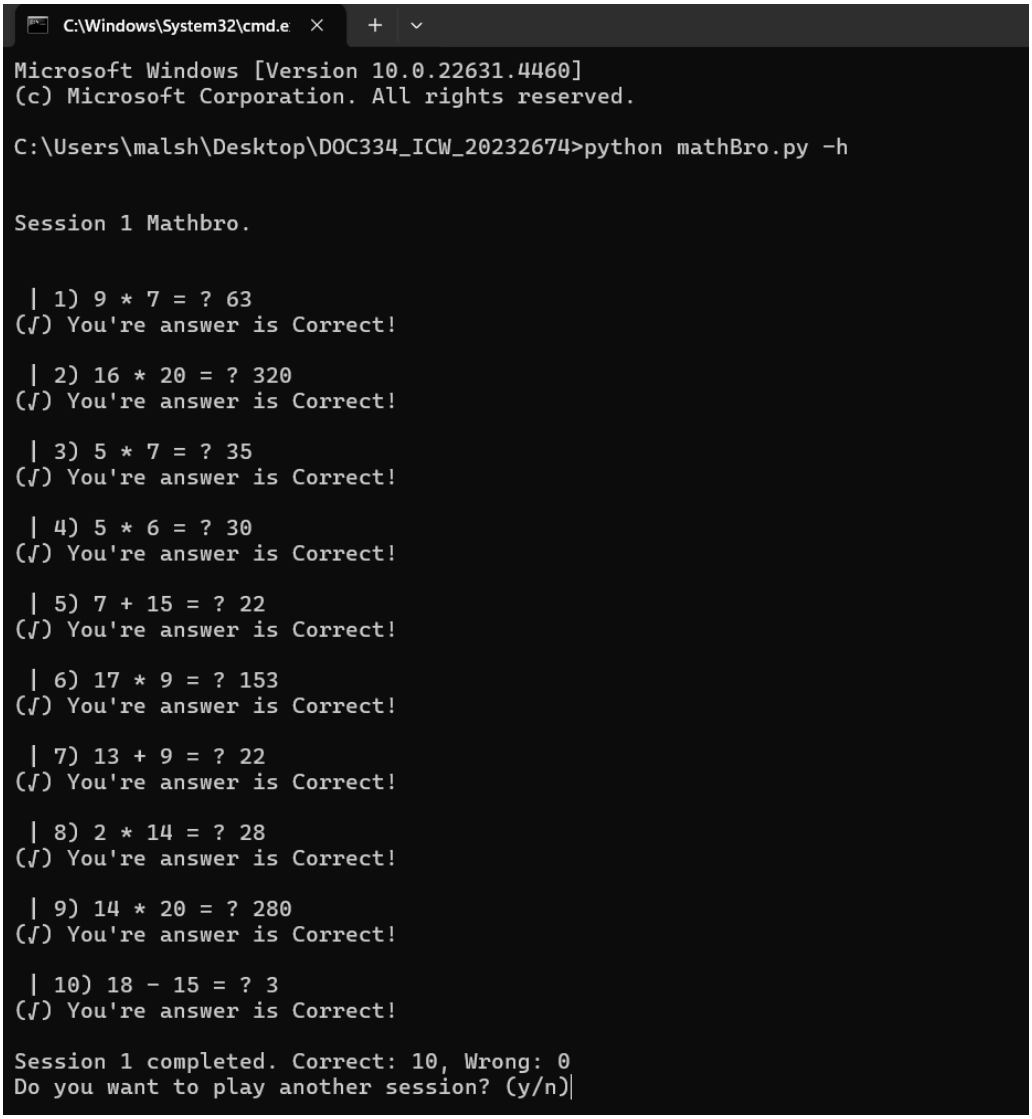


*Figure 14.Correct Input - Hard*

### 5.4.2 Input Correct answers , Wrong answers

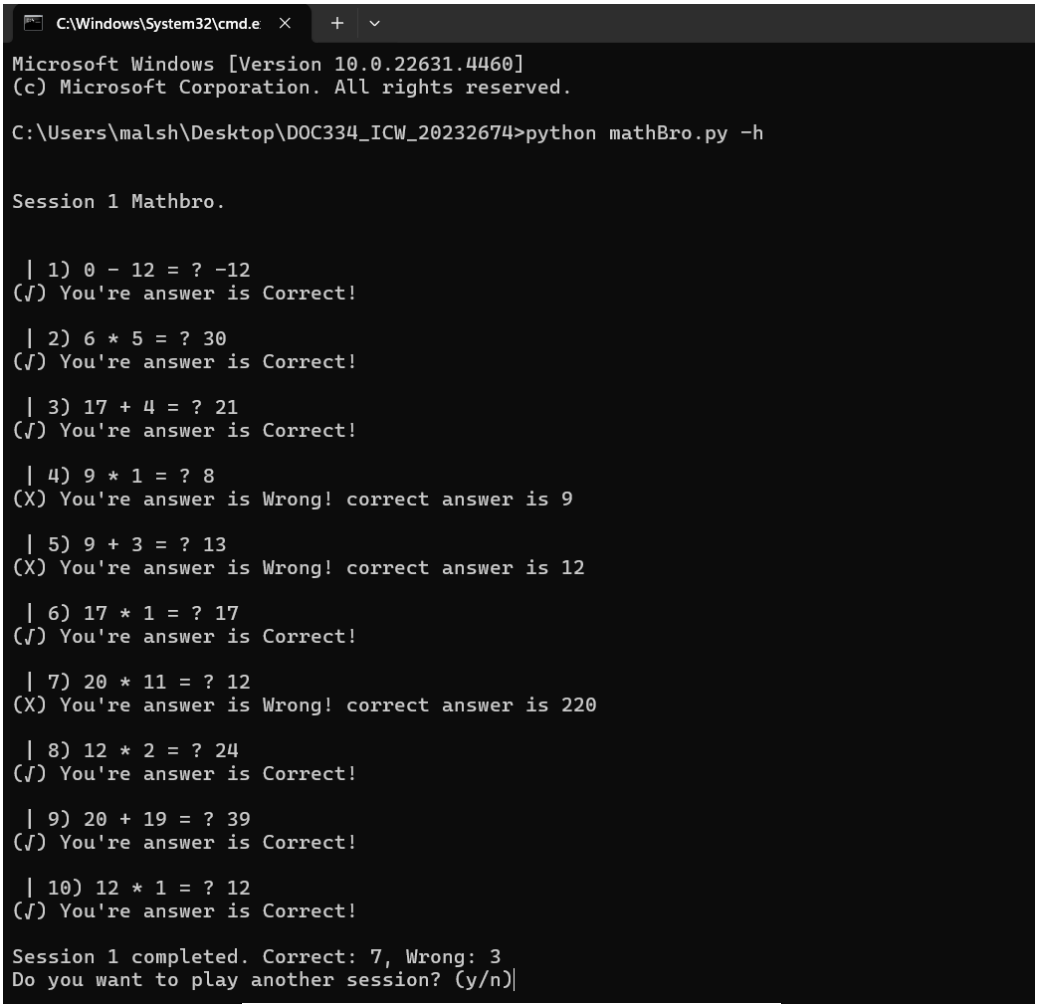


*Figure 15.Correct Input & Wrong Input - hard*

### 5.4.3 Input String

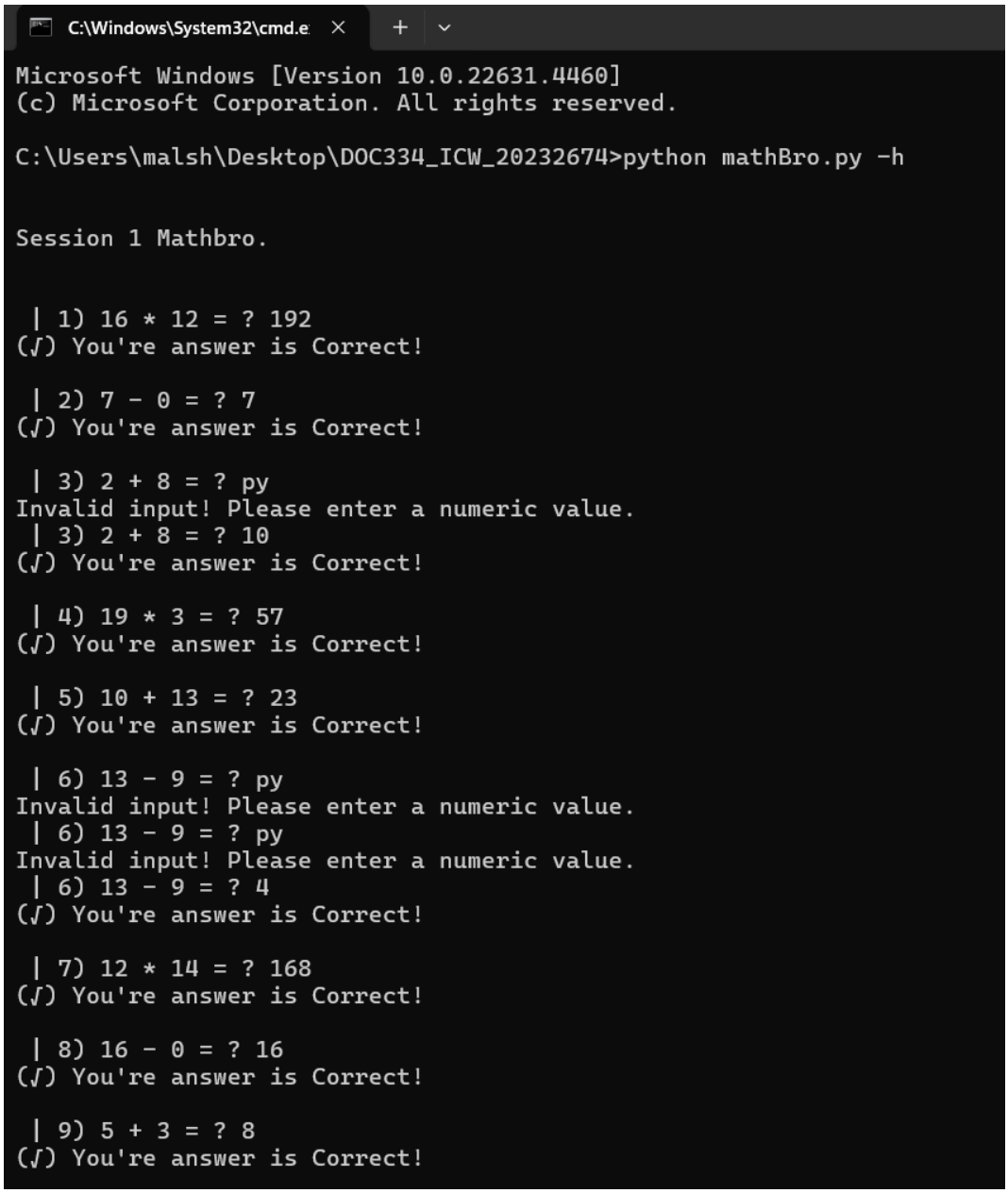


*Figure 16.Input String - Hard*

### 5.4.4 Play another session

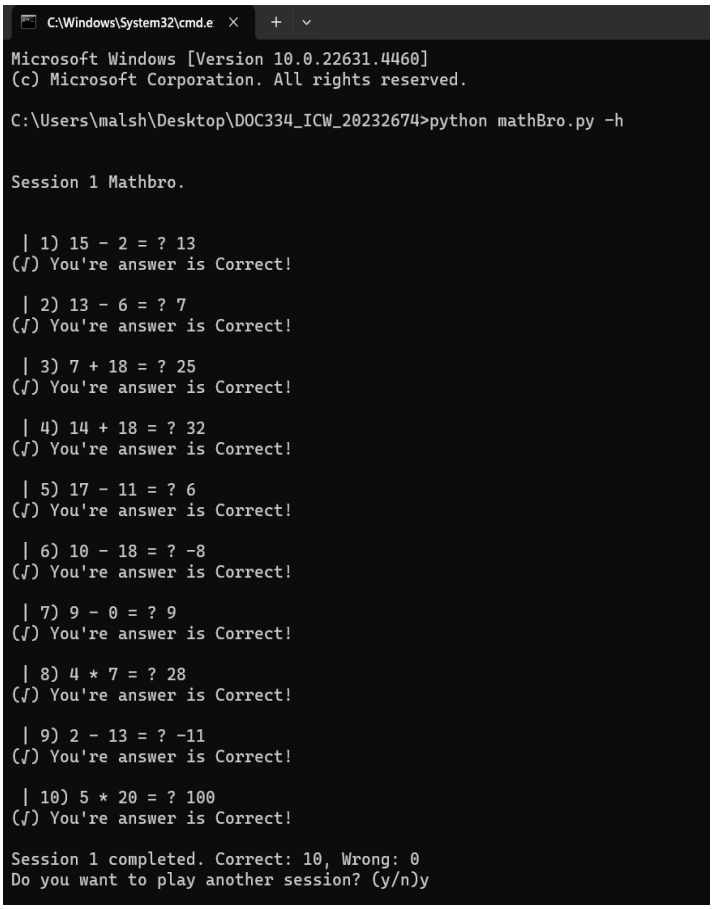


*Figure 18.Play another session1 - Hard*

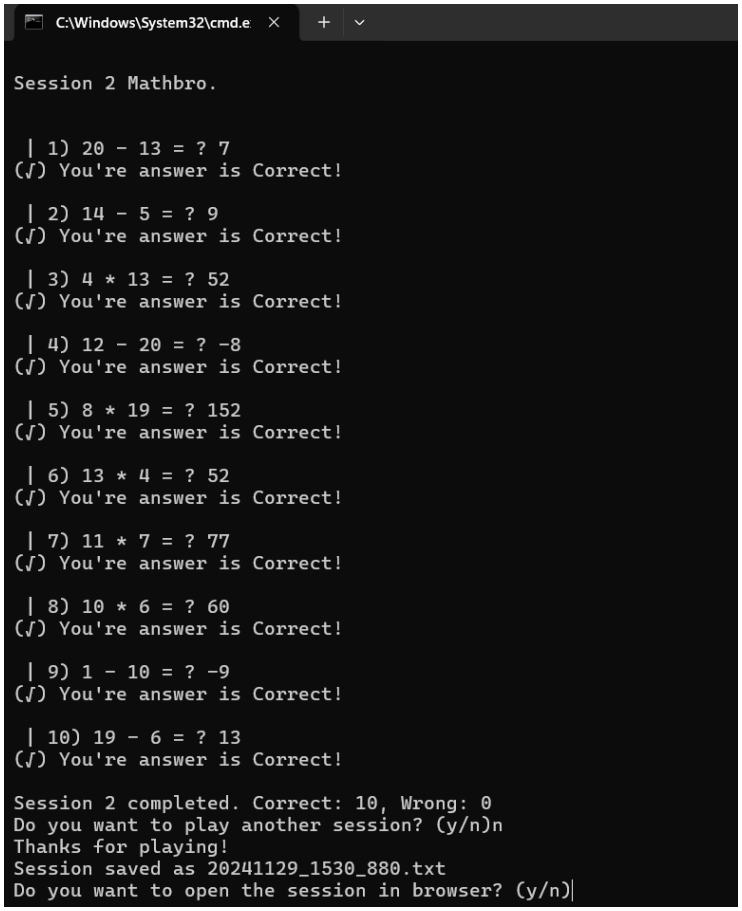


*Figure 17.Play another session2 -Hard*

# 6.Text file Screenshots

## 6.1 Demo Mode

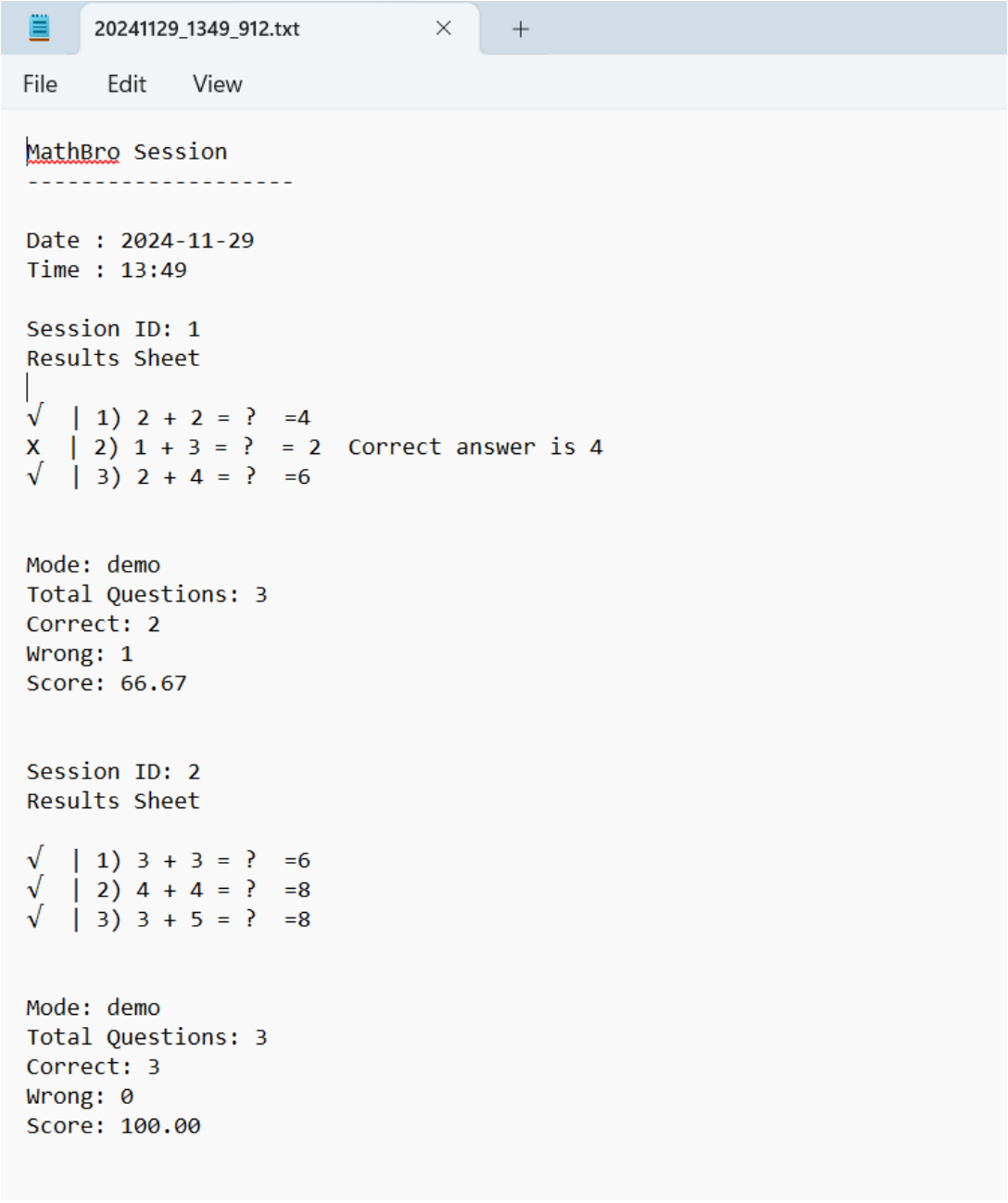


*Figure 19.Text-Demo*

## 6.2 Easy Mode

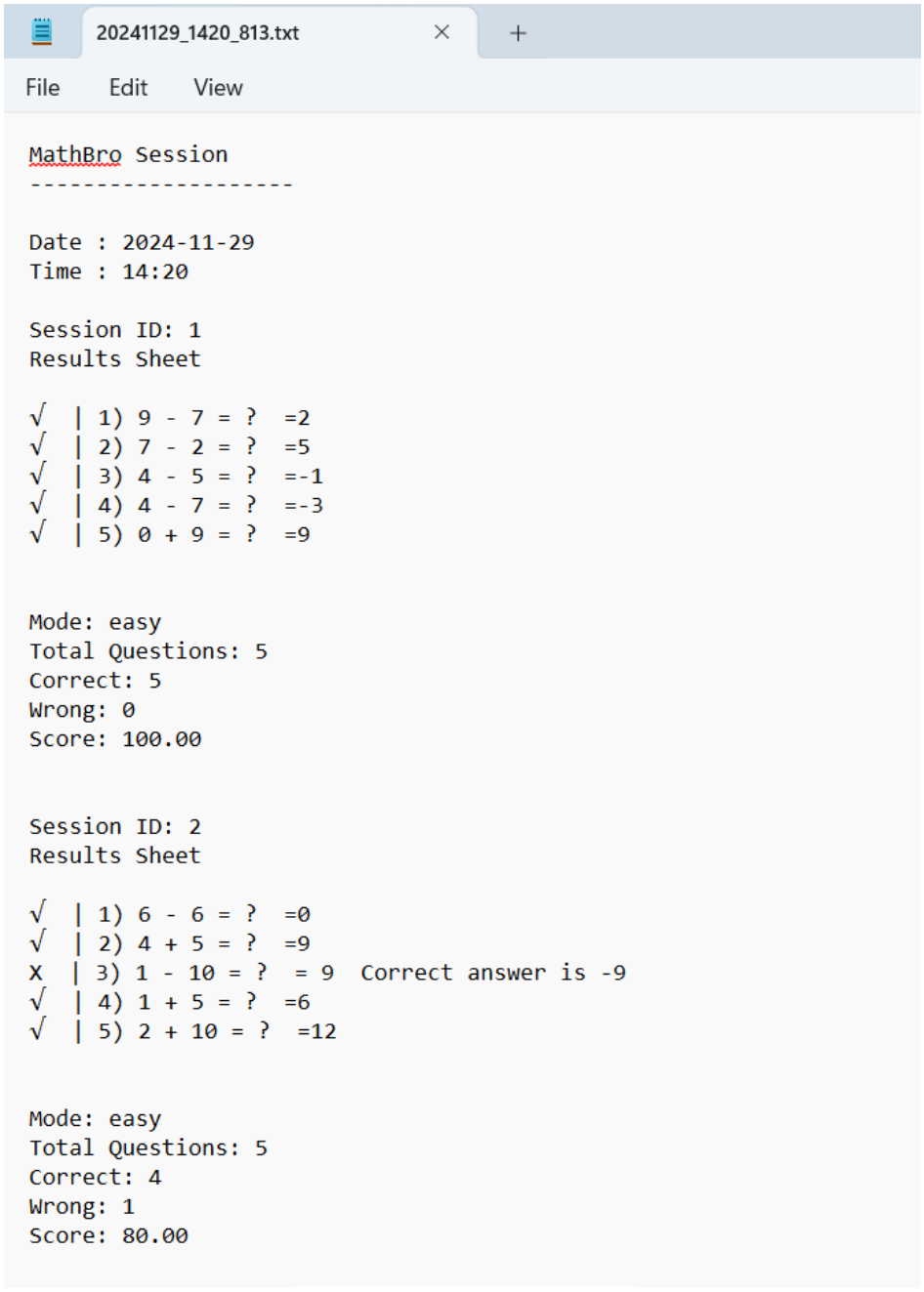


*Figure 20.Text -Easy*

## 6.3 Medium Mode

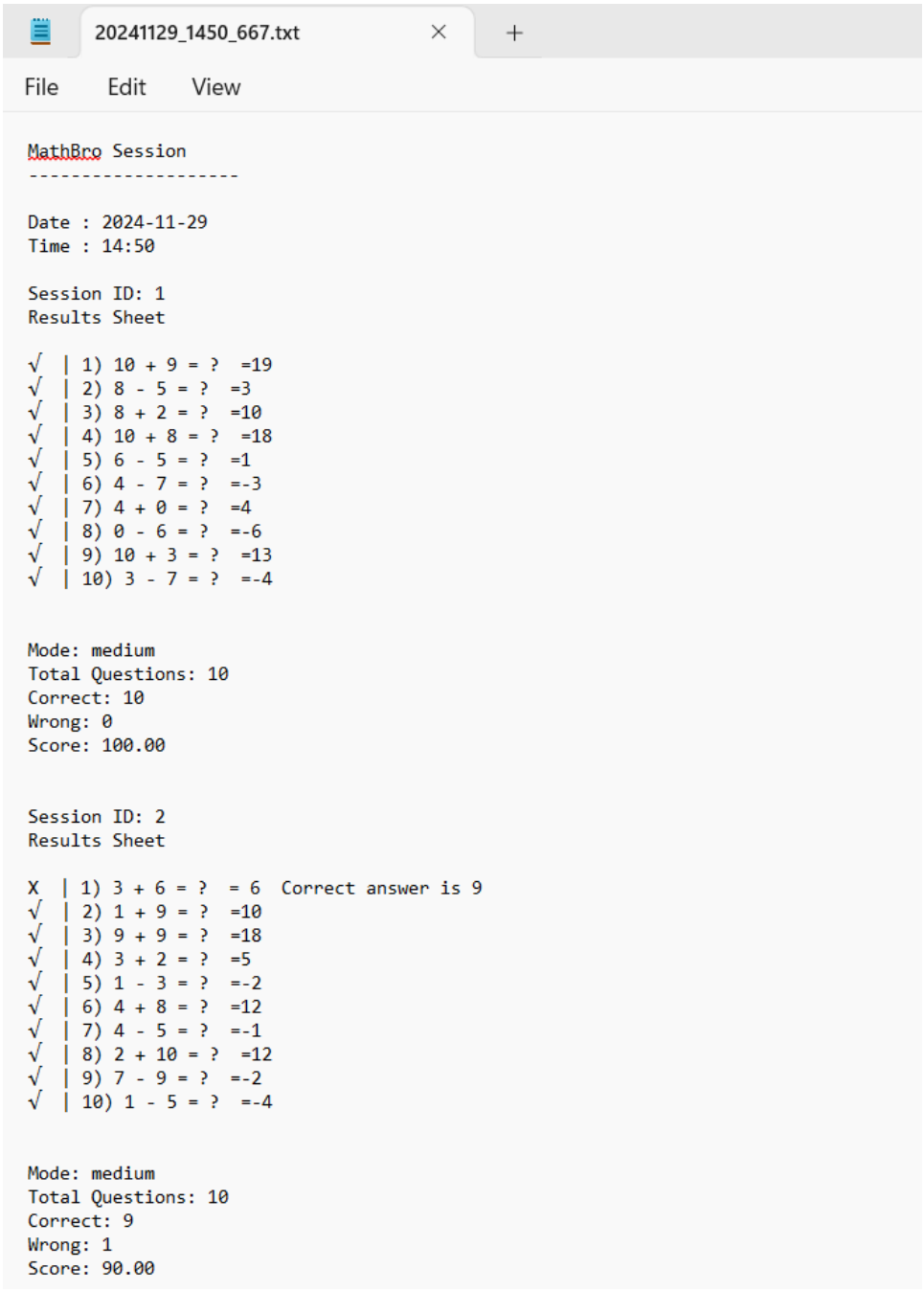


*Figure 21.Text - Medium*

## 6.4 Hard Mode

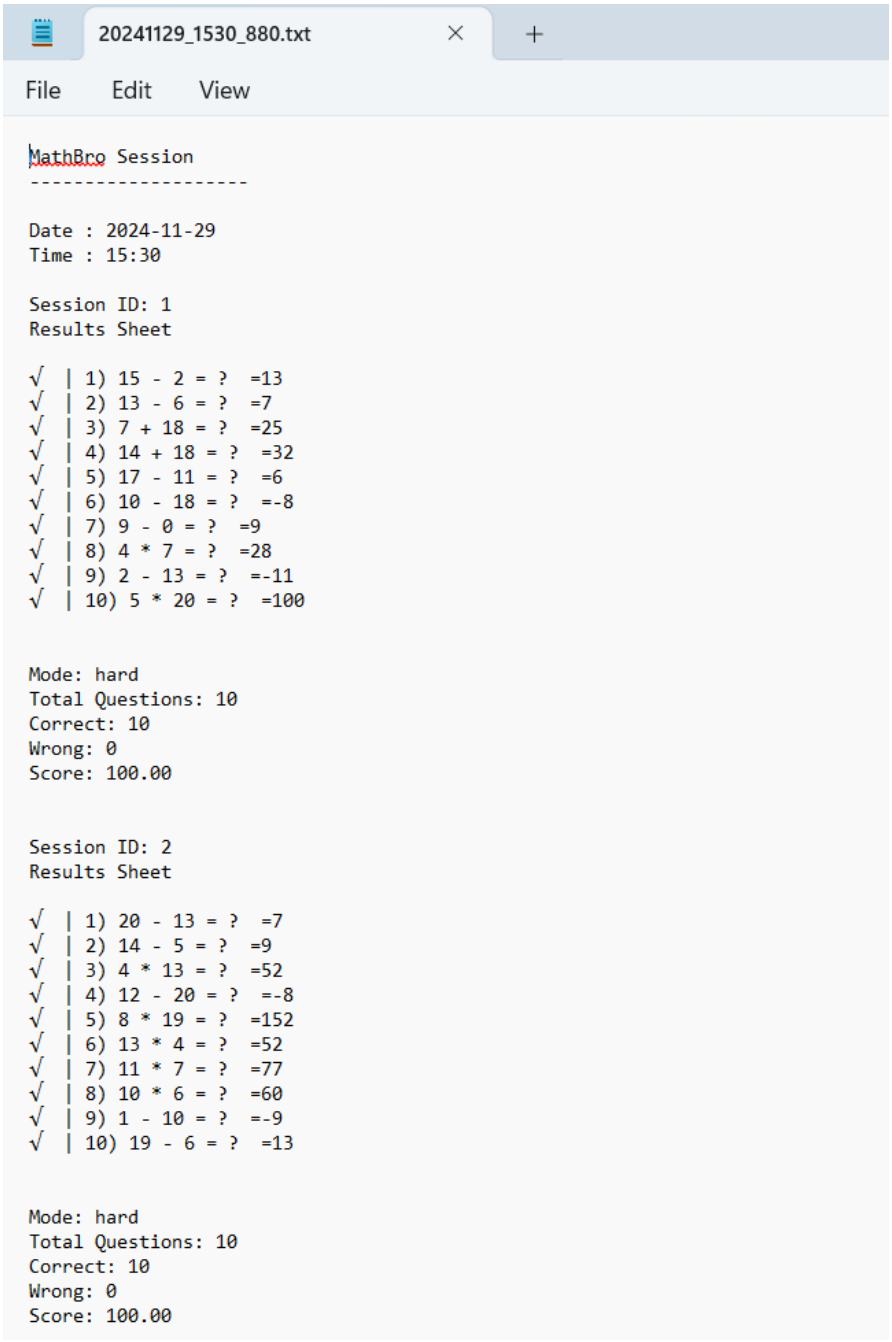


*Figure 22.Text - Hard*

# 7.HTML file Screenshots

## 7.1 Demo Mode


*Figure 23.Html1 - Demo*

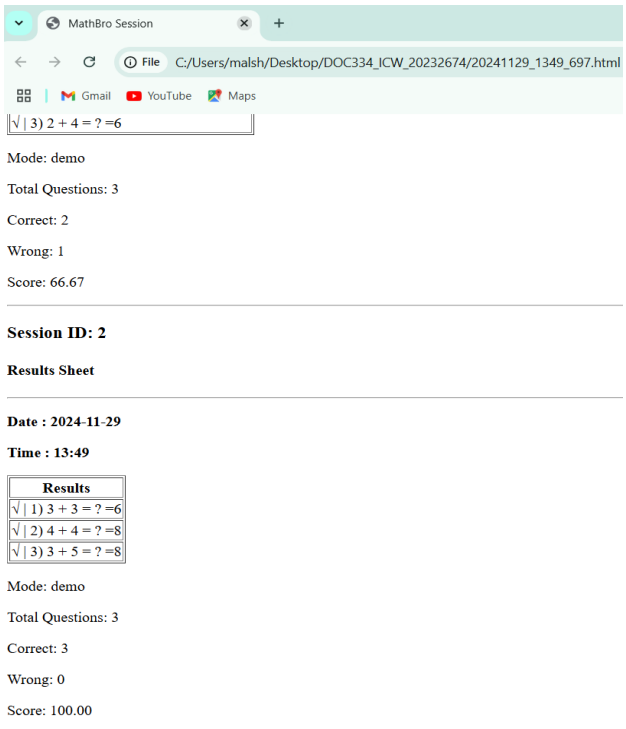


*Figure 24.Html2 - Demo*

## 7.2 Easy Mode

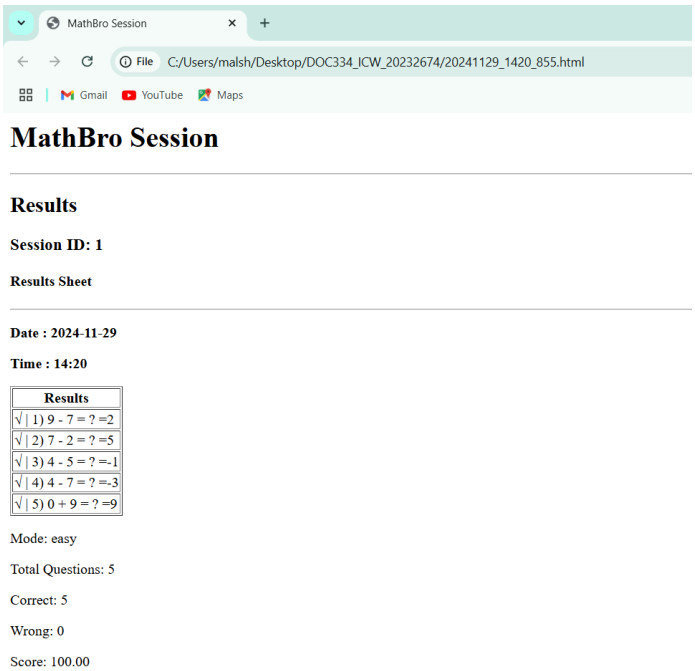


*Figure 25.Html1 - Easy*

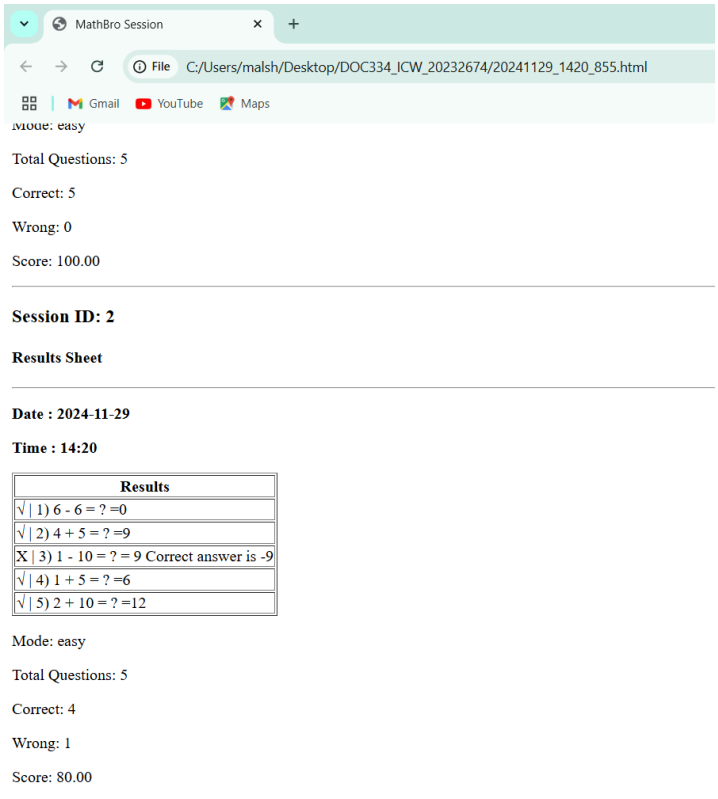


*Figure 26.Html2 -Easy*

## 7.3 Medium Mode

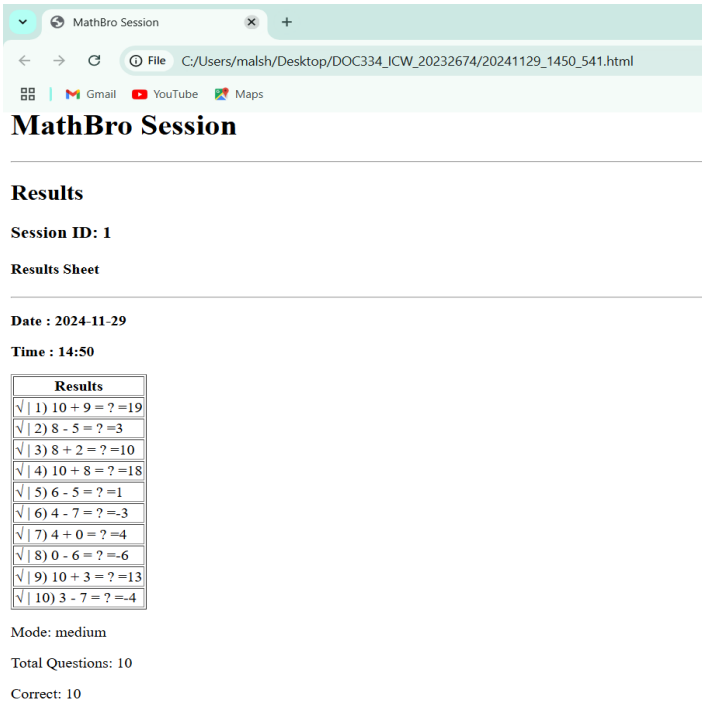


*Figure 27.Html1 - Medium*

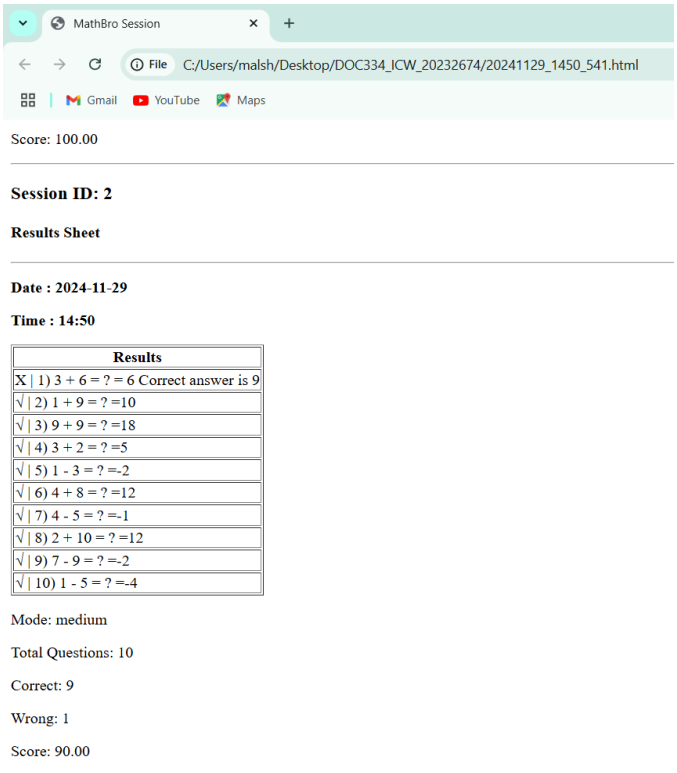


*Figure 28.Html2 - Medium*

## 7.4 Hard Mode

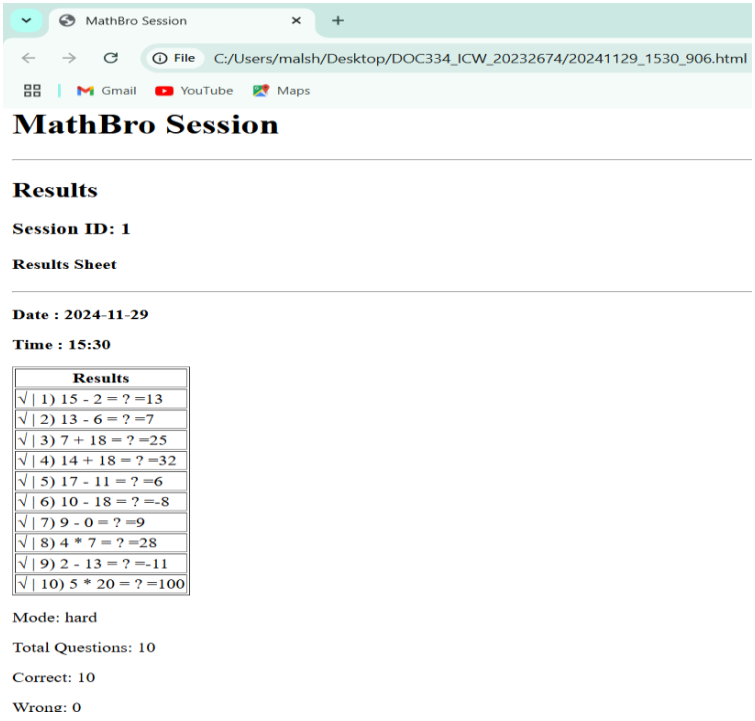


*Figure 29.Html1 - Hard*

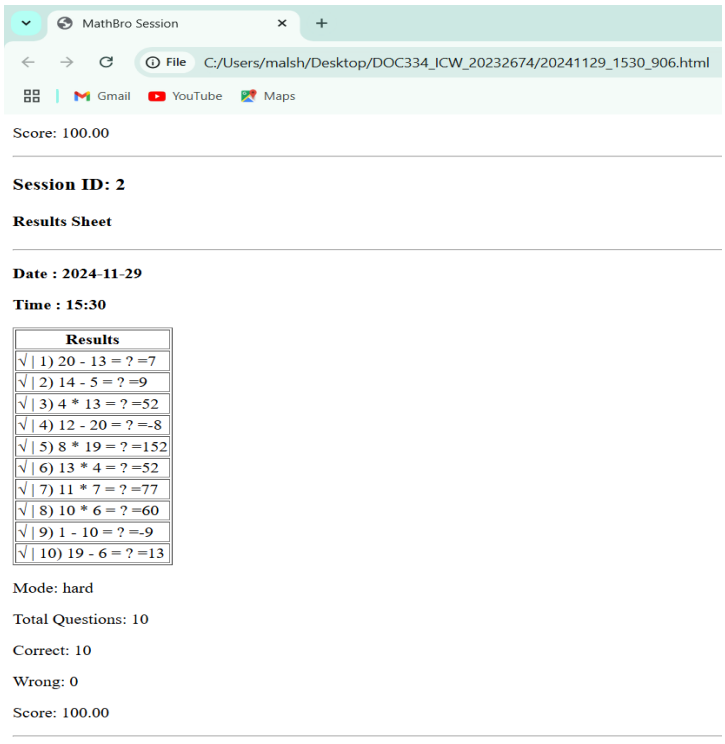


*Figure 30.Html2 - Hard*

# 8.Test Cases

## 8.1 Test Case 1 :-

| Test Case No | Input Entered | Expected output | Actual output | Result |
|---|---|---|---|---|
| 01 | python mathBro.py -e | Mode : Easy<br>Questions : 5<br>Range : 0-10<br>Operators : + , - | Mode : Easy<br>Questions : 5<br>Range : 0-10<br>Operators : + , - | Pass |

*Table 1.Test Case 1*

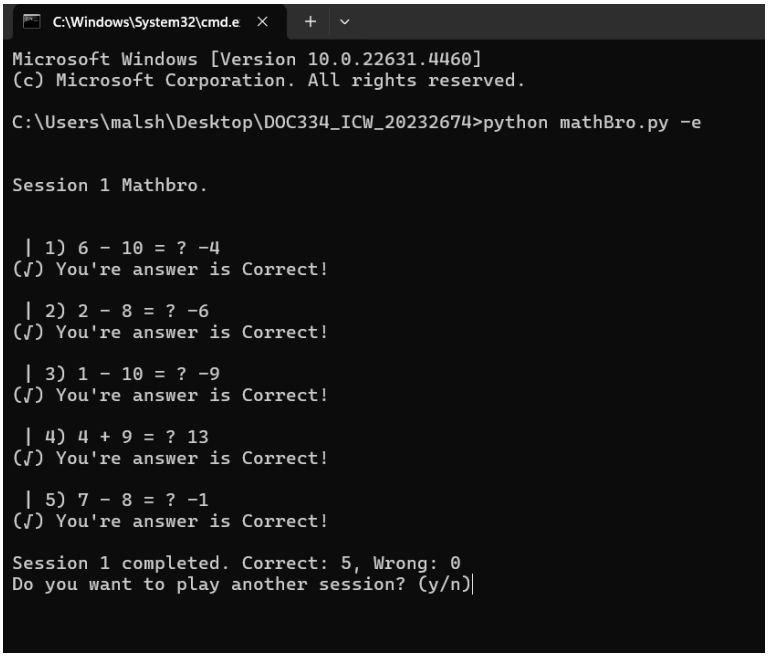


*Figure 31.Test case1*

## 8.2 Test Case 2 :-

| Test Case No | Input Entered | Expected output | Actual output | Result |
|---|---|---|---|---|
| 02 | python mathBro.py -x | Prints "Invalid mode selected." and exits | Prints "Invalid mode selected." and exits | Pass |

*Table 2.Test Case 2*

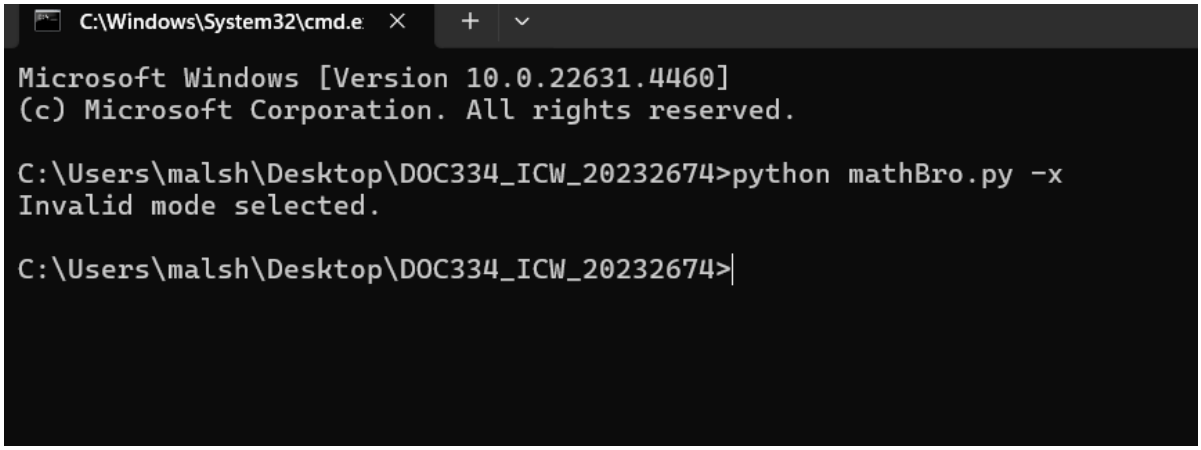


*Figure 32.Test case2*

## 8.3 Test Case 3 :-

| Test Case No | Input Entered | Expected output | Actual output | Result |
|---|---|---|---|---|
| 03 | User enters **“16”** for 8 + 8 | "(√) Your answer is Correct!" | "(√) Your answer is Correct!" | Pass |

*Table 3.Test Case 3*

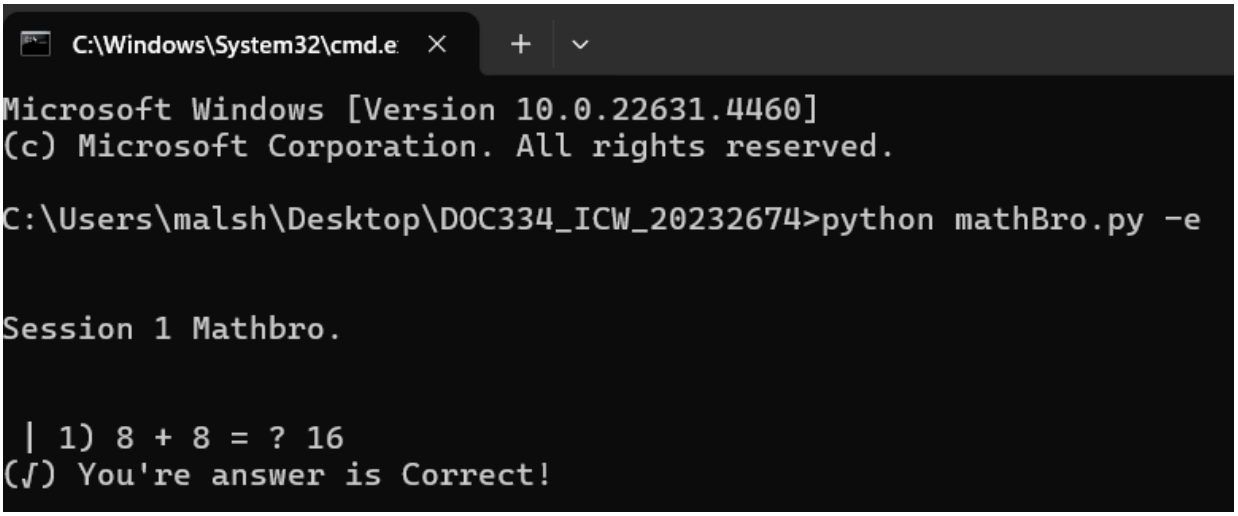


*Figure 33.Test Case 3*

## 8.4 Test Case 4 :-

| Test Case No | Input Entered | Expected output | Actual output | Result |
|---|---|---|---|---|
| 04 | User enters **“0”** for 17 - 0 | "(X) Your answer is Wrong! Correct answer is 17" | "(X) Your answer is Wrong! Correct answer is 17" | Pass |

*Table 4.Test Case 4*

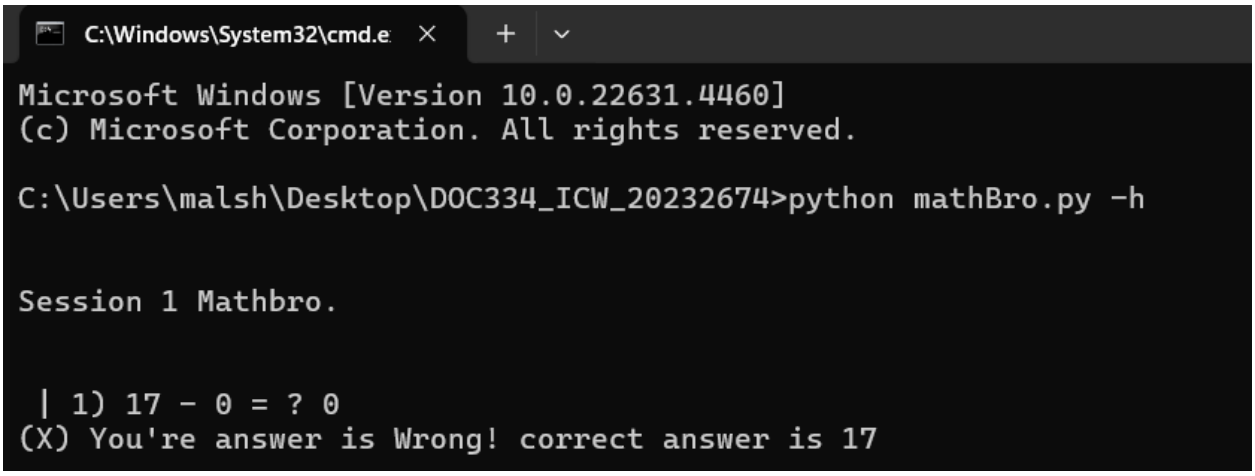


*Figure 34.Test Case 4*

## 8.5 Test Case 5 :-

| Test Case No | Input Entered | Expected output | Actual output | Result |
|---|---|---|---|---|
| 05 | User enters "**py**" | "Invalid input! Please enter a numeric value." | "Invalid input! Please enter a numeric value." | Pass |

*Table 5.Test Case 5*

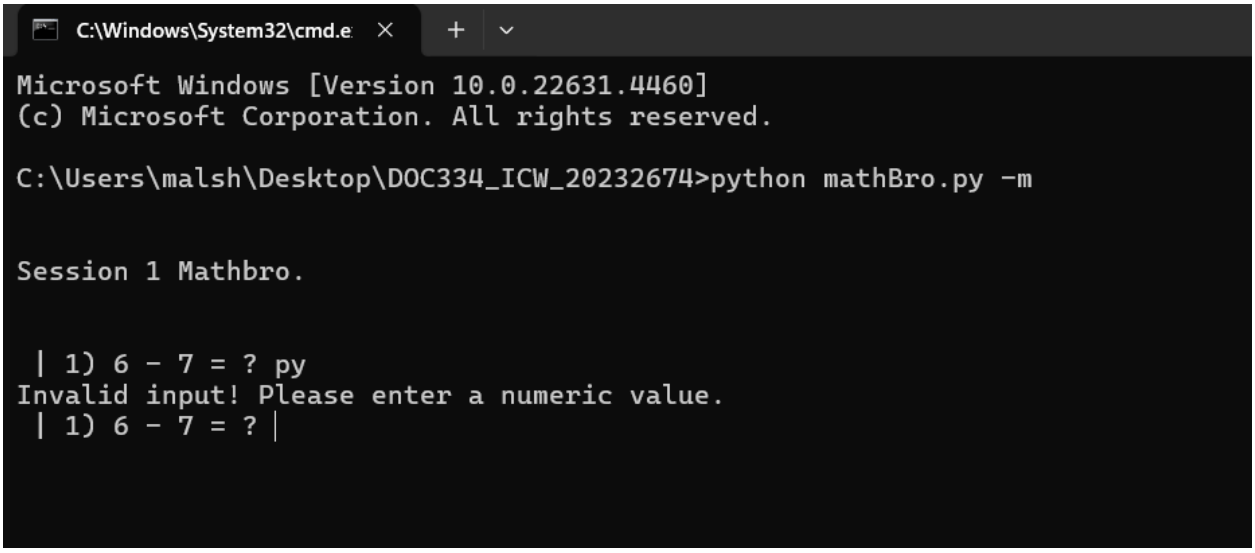


*Figure 35.Test Case 5*

## 8.6 Test Case 6 :-

| Test Case No | Input Entered | Expected output | Actual output | Result |
|---|---|---|---|---|
| 06 | User enters **“y”** to open HTML | HTML file is generated and opens in browser | HTML file is generated and opens in browser | Pass |

*Table 6.Test Case 6*

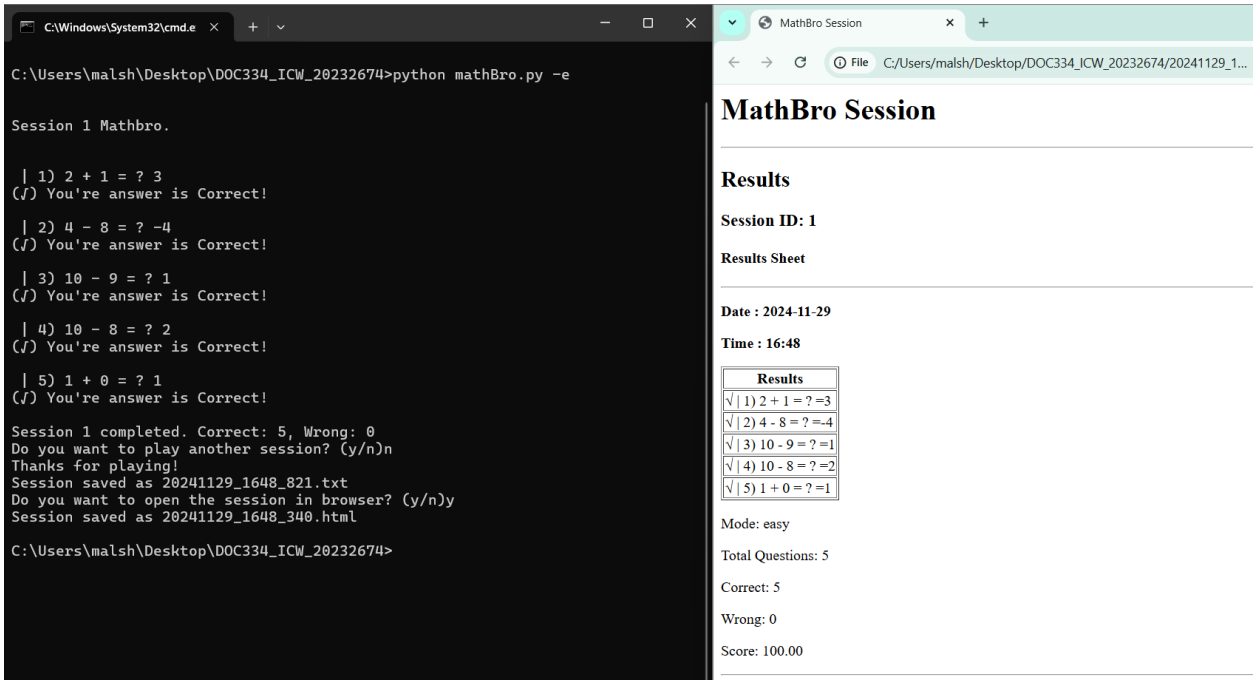


*Figure 36.Test Case 6*

## 8.7 Test Case 7 :-

| Test Case No | Input Entered | Expected output | Actual output | Result |
|---|---|---|---|---|
| 07 | Do you want to play another session? **“n”** | Print “Thanks for playing!) and File created with session details | Print “Thanks for playing!) and File created with session details | Pass |

*Table 7.Test Case 7*

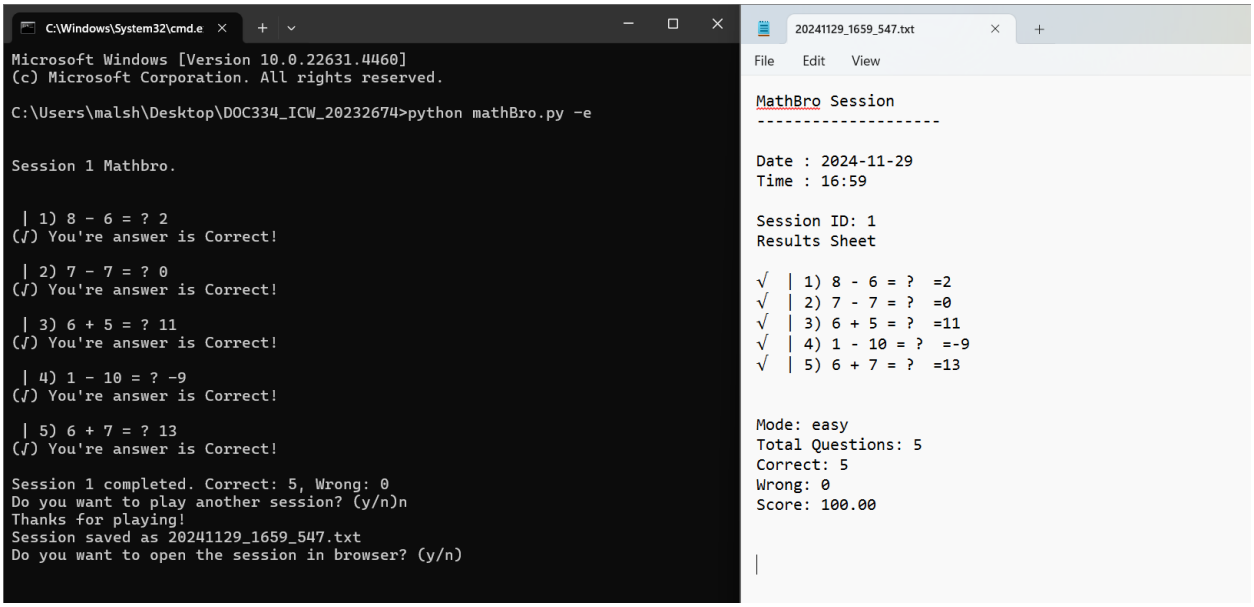


*Figure 37.Test Case 7*

## 8.8 Test Case 8 :-

| Test Case No | Input Entered | Expected output | Actual output | Result |
|---|---|---|---|---|
| 08 | Complete 2 consecutive sessions, input **“y”** to continue | Both sessions saved and displayed separately | Both sessions saved and displayed separately | Pass |

*Table 8.Test Case8*

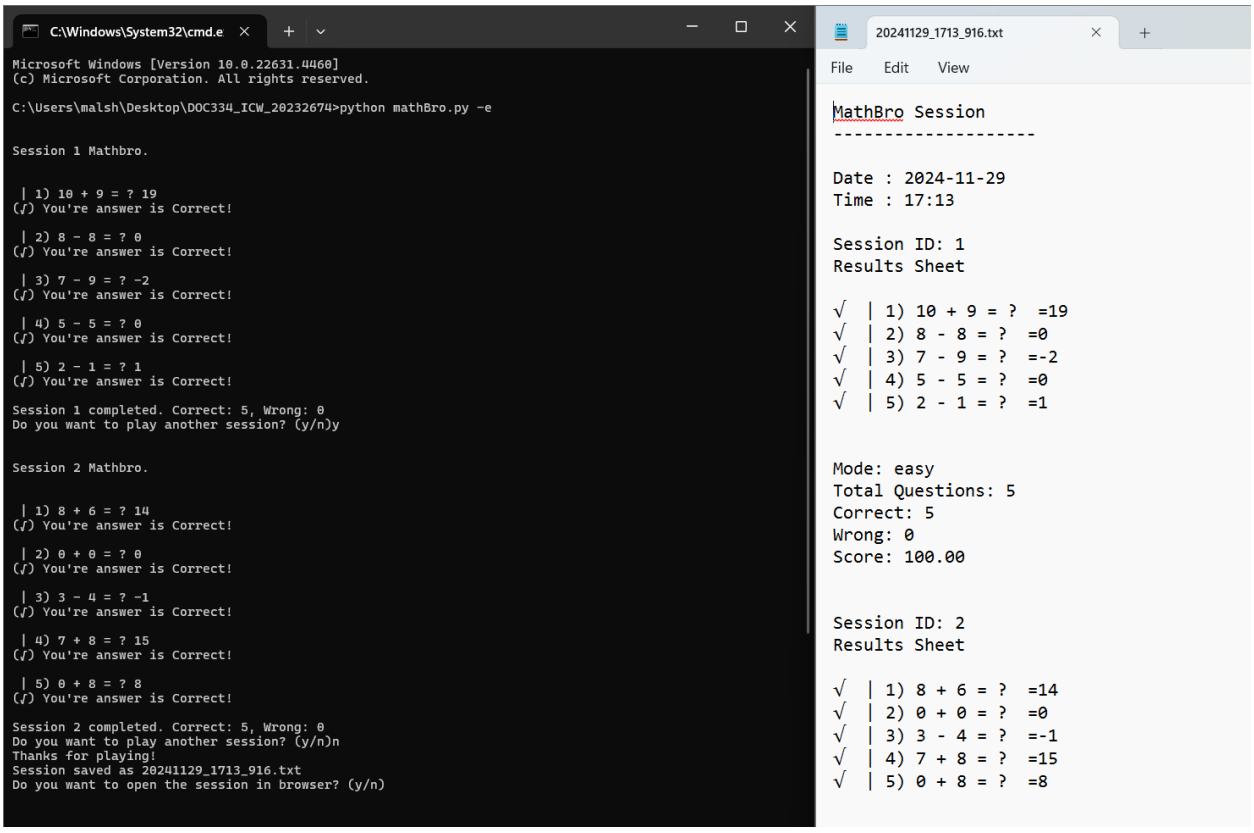


*Figure 38.Test Case 8*

## 8.9 Test Case 9 :-

| Test Case No | Input Entered | Expected output | Actual output | Result |
|---|---|---|---|---|
| 09 | 5 correct out of 10 questions | Score: 50% | Score: 50% | Pass |

*Table 9.Test Case 9*

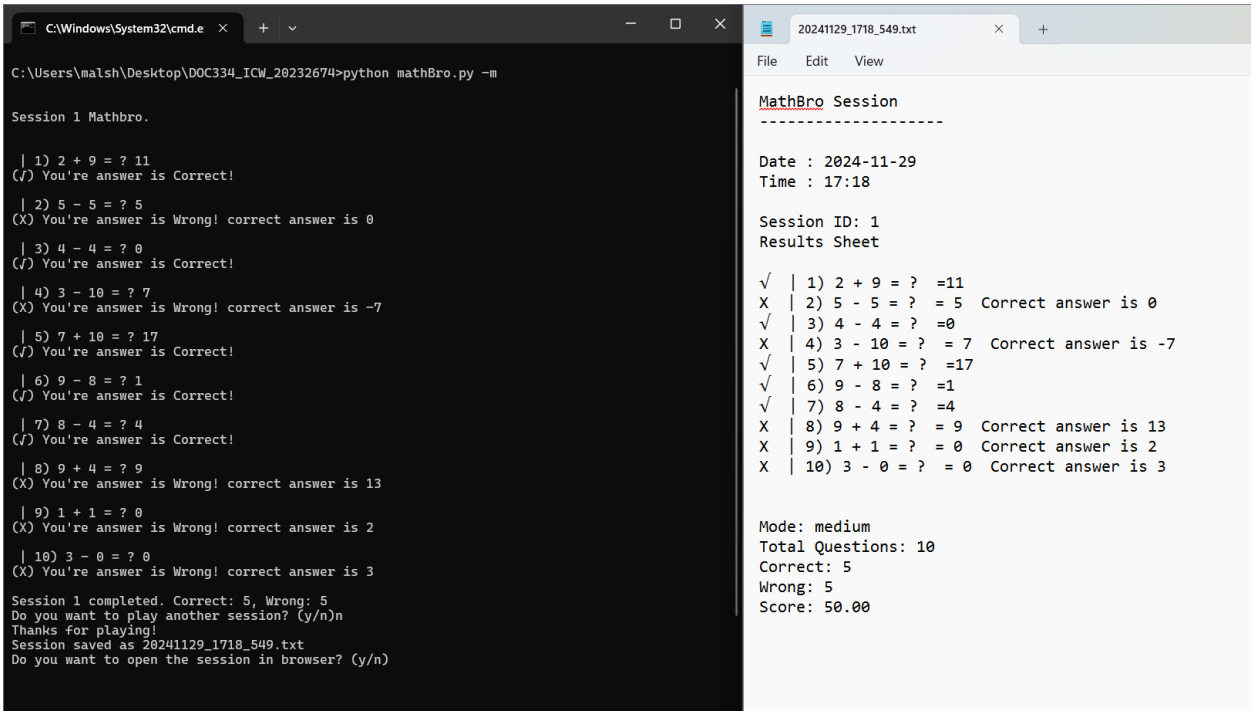


*Figure 39.Test Case 9*

## 8.10 Test Case 10 :-

| Test Case No | Input Entered | Expected output | Actual output | Result |
|---|---|---|---|---|
| 10 | No command-line argument | Mode : demo<br>Questions : 3<br>Range : 0-5<br>Operators : + | Mode : demo<br>Questions : 3<br>Range : 0-5<br>Operators : + | Pass |

*Table 10.Test Case 10*

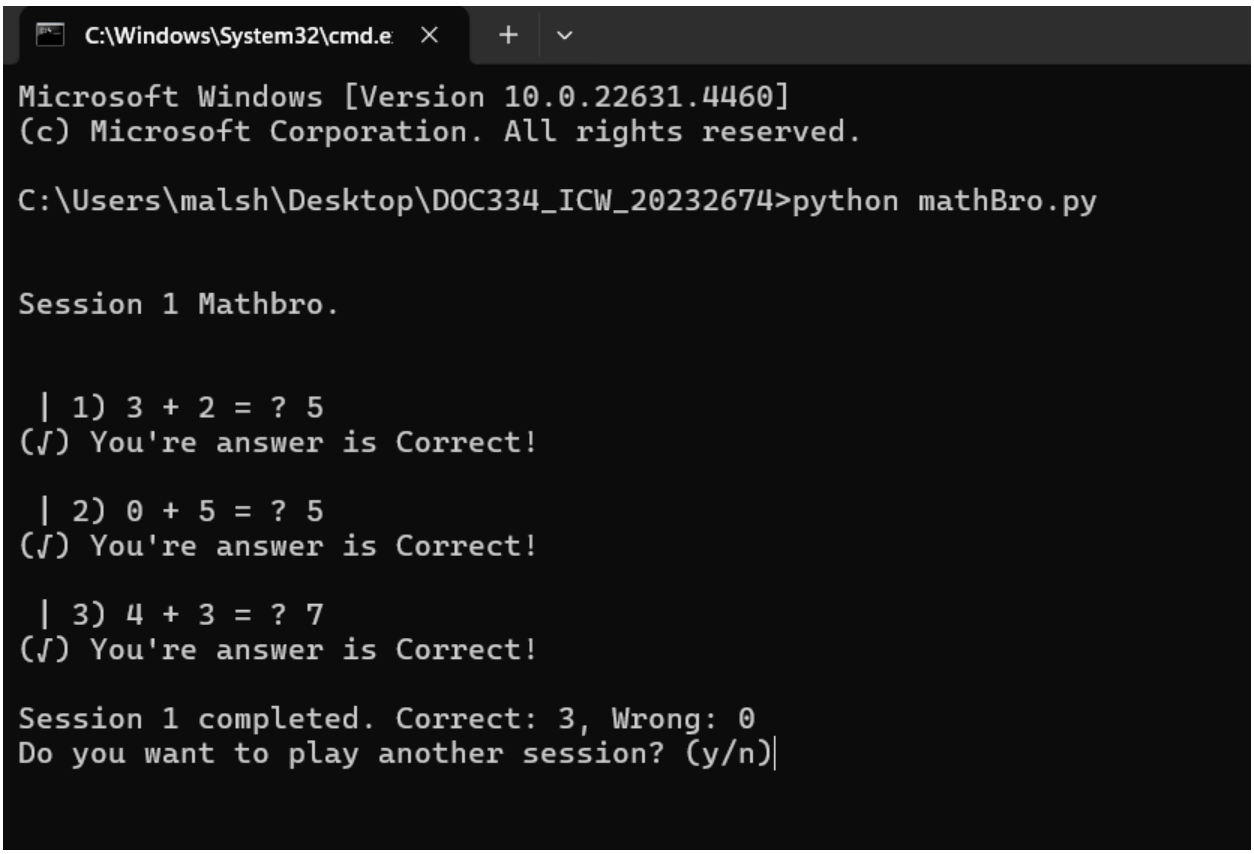


*Figure 40.Test Case 10*

# 9. Conclusion

Arithmetic operations-based interactive math quiz game with four difficulty levels (Demo, Easy, Medium, and Hard) is available through the MathBro application. In addition to getting instant feedback and solving randomly generated questions, users can monitor their progress via text and HTML reports. By allowing outcomes to be kept and examined, encouraging continued practice, and offering a thorough session summary, the application successfully raises learning engagement. All things considered, MathBro is a useful educational resource for getting better at math in an enjoyable and adaptable method.