**Artificial Intelligent (Lab)**

**Task # 02**

****

**Submitted To: Sir Rasikh Ali**

**Submitted By: Shumaila Maryam**

**Roll no: SU92-BSDSM-F24-062**

**Submitted On: 3-September-2025**

**Section: BSDS-3A**

**Department of Software Engineering, Superior University, Lahore**

**Movie Budget Analysis Program**

**Introduction:**

The Movie Budget Analysis Program is a simple yet effective tool designed to analysed the financial data of movies. It takes a list of movies along with their budgets, allows the user to add new movies, and then performs a comparison of each movie’s budget against the overall average. The program provides detailed insights into which movies are above or below average in terms of budget.

**Why I Made This Program:**

The main idea behind this program is to practice and demonstrate concepts of lists, loops, user input, and conditional statements in Python. Movies were chosen as the theme because they are relatable, making the program engaging while still being educational. By making this program, I wanted to:

* Learn how to work with collections of data (lists of movies).
* Understand how to calculate totals and averages.
* Gain experience in taking user input and dynamically updating data.
* Develop logic for comparing values and drawing meaningful conclusions.

This program serves both as a learning project and a practical analysis tool, helping to strengthen programming skills.

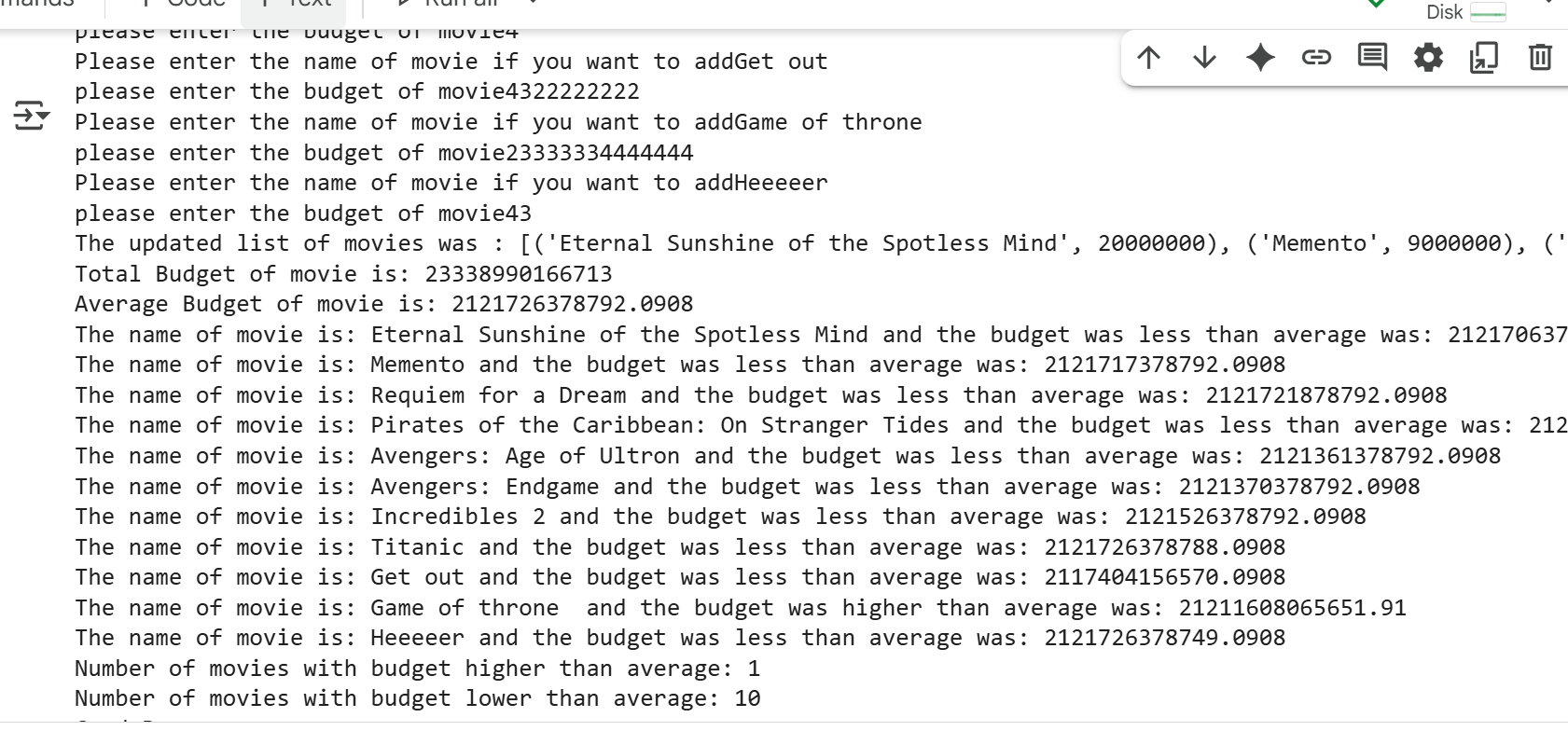
**How It Works:**

1. **Starting with Data**  
   The program begins with an existing list of movies and their budgets. This ensures that analysis can start immediately.
2. **Adding New Movies**  
   The user is asked how many movies they want to add. For each movie, the program records its name and budget, then adds it to the list. This makes the program flexible and interactive.
3. **Updated Dataset**  
   Once the new movies are added, the program displays the complete updated list of movies. This reassures the user that their input was successfully recorded.
4. **Budget Calculations**  
   The program calculates the total budget of all movies combined, as well as the average budget across the dataset. These values act as the basis for further analysis.
5. **Comparisons**  
   Each movie’s budget is compared with the average:
   * If a movie’s budget is higher than average, the program shows by how much.
   * If a movie’s budget is lower than average, the program shows the difference.
   * If a movie’s budget is exactly equal to the average, that is highlighted as well.
6. **Counting Results**  
   The program also keeps track of how many movies are above the average and how many are below. This provides a quick summary of the dataset.
7. **Final Output**  
   At the end, the program prints a clear summary of results and displays a friendly message to close.

**Summary**

1. The Movie Budget Analysis Program is a Python project designed to analyze movie budgets. It starts with a list of movies, allows the user to add new ones, and then calculates the total and average budgets. Each movie is compared against the average to show whether its budget is higher, lower, or equal. The program also counts how many movies fall above or below the average.
2. This project was created to practice Python concepts such as lists, loops, user input, and conditional logic while applying them to a relatable topic. It is both an educational exercise and a practical tool for simple data analysis.

**Output:**



**Fizz Buzz Game**

**Introduction:**

The FizzBuzz game is a fun, interactive number game where the computer generates random numbers, and the player guesses whether the number should be replaced by Fizz, Buzz, FizzBuzz, or simply remain a number.This version is coded in Python using Object-Oriented Programming (OOP) principles. It also keeps track of the player’s score, providing instant feedback after every round.

**Why I made this Program:**

This program was created as a learning project to practice:

* Using classes and objects in Python.
* Applying loops and conditions effectively.
* Keeping track of scores and showing results in real-time.
* Making programming practice interactive and engaging through a game format.

The FizzBuzz problem itself is a classic coding exercise, often asked in interviews, and this version extends it by adding score calculation, random numbers, and continuous rounds.

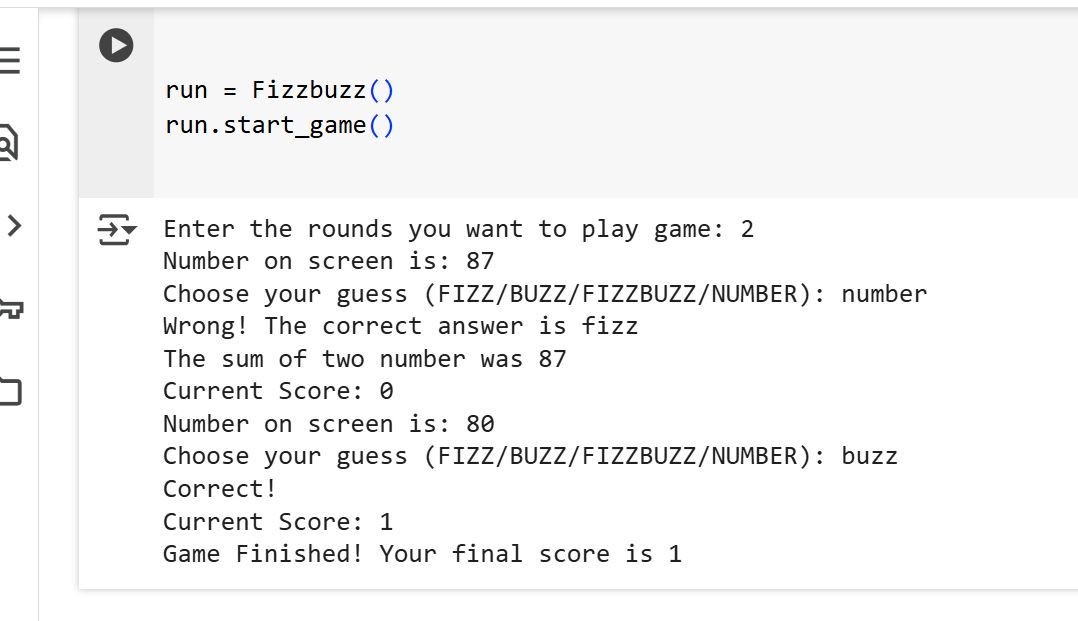
**How the Program Works:**

1. **Class Setup:**
   * When the game starts, two values are initialized:
     + First\_number stores the previous random number (used for sum calculation).
     + score keeps track of the player’s points.
2. **Check the Number:**
   * The computer checks the number with these rules:
     + If divisible by both 3 and 5, the correct answer is FizzBuzz.
     + If divisible by only 3, the answer is Fizz.
     + If divisible by only 5, the answer is Buzz.
     + Otherwise, the answer is Number.
3. **Starting the Game** 
   * The player decides how many rounds to play.
   * For each round:
     + A random number (between 1–100) is shown on the screen.
     + The player guesses **Fizz, Buzz, FizzBuzz, or Number**.
     + The program checks if the guess is correct using check\_game().
4. **Correct vs Wrong Guess**
   * If correct → the player’s score increases by +1.
   * If wrong → the program shows the correct answer and also calculates the sum of the previous number and the current number (just as an extra challenge).
     + The score decreases by 1 (only if it’s above 0).
5. **Updating After Each Round**
   * After every round:
     + First\_number is updated to the current number.
     + The current score is shown instantly to keep the game interactive.
6. **End of Game**
   * Once all rounds are finished, the program prints the final score.

**Summary:**

FizzBuzz is a counting game where numbers divisible by 3 are replaced with Fizz, divisible by 5 with "Buzz", and divisible by both with FizzBuzz.  
Players can also add the previous and current number and guess the next correct output, adding a mental challenge.  
It tests basic programming logic like loops, conditionals, and simple arithmetic.

**Output:**

****