

**Name: Aliza Malik**

**Roll: Su92BSAIM051**

**Sir: Rasikh Ali**

**AI LAB TASK :2**

**Mini Project :1**

FizzBuzzGame Documentation

Introduction

FizzBuzzGame is a simple Python-based game that tests the user's understanding of the classic FizzBuzz logic. The game runs for a set number of rounds, where the computer displays a number, and the user has to respond with the correct FizzBuzz answer.However, there’s a twist: instead of responding to the number shown by the computer, the user has to respond to the **sum of the current number and the previous number**. This makes the game slightly more challenging and keeps the user alert.

Main Functions

**1. Constructor (\_\_init\_\_)**  
Initializes the game. It sets the total number of rounds and tracks the previously chosen number.

**2. fizzbuzz\_rule()**  
Decides what the correct FizzBuzz output is for any given number:

* If divisible by 3 → returns “Fizz”
* If divisible by 5 → returns “Buzz”
* If divisible by both 3 and 5 → returns “Fizz Buzz”
* Otherwise, returns the number itself as a string

**3. play()**  
Runs the game loop. For each round:

* The computer picks and displays a number
* The user must respond based on the sum of the current and previous number
* The input is checked against the correct FizzBuzz answer
* Feedback is given (correct or wrong)
* The previous number is updated for the next round

Problems and Solutions

**Problem 1: The user doesn't know which number to respond to**  
The actual number the user should respond to (sum of current and previous) is not shown.

**Solution:** Clearly display the number the user needs to answer for in each round.

**Problem 2: The first round is confusing**  
In the first round, the previous number is zero, which can make the sum seem unclear.

**Solution:** Add a short note in the instructions explaining that the first round is based only on the first number, and from the second round onward, the sum will be used.

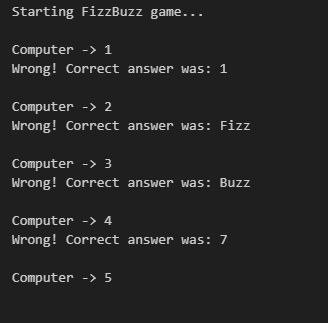
**Problem 3: Input formatting issues**  
If the user enters something like “ Fizz ” or uses capital letters, the input may be marked wrong even if the answer is correct.

**Solution:** Inputs are already being trimmed and converted to lowercase before checking, which solves the problem. Just ensure consistency in comparison logic.

Features

* Customizable number of rounds
* Follows standard FizzBuzz rules with a twist
* Real-time feedback on each answer
* Keeps track of the previous number to calculate the next expected answer
* Ignores case sensitivity and extra spaces in user input

This game adds a small challenge to the usual FizzBuzz logic and is good for practicing basic logic, input handling, and interactive Python programming. You can also expand it by adding features like scoring, difficulty levels, or even a timer.



MINI PROJECT 2

Introduction

The Movie Budget Analysis Project is a Python program that focuses on analyzing movie budgets. Its main purpose is to calculate the **average budget** of all movies and identify which ones are above this average. The project is designed using **object-oriented programming** and includes error handling for smooth user interaction. It also gives users the flexibility to **add their own movies**, making the analysis more dynamic and practical.

Features

* Built using **Object-Oriented Programming** for better structure
* Allows users to **add movies dynamically** with validation
* Calculates the **average budget** of all movies
* Finds movies that **exceed the average budget** and shows by how much
* Produces a **final report** with clear results

**Functions Used**

* **\_\_init\_\_** → Initializes objects and converts input data into structured movie objects
* **\_\_repr\_\_** → Returns a readable representation of a movie (title + budget)
* **add\_movies()** → Lets users add new movies with proper input checks
* **calculate\_average\_budget()** → Computes the overall average budget
* **find\_high\_budget\_movies(average)** → Identifies movies above the average
* **report()** → Generates and displays the final summary report

Problems Faced and Solutions

1. **Invalid User Input** → Users entered text instead of numbers for budgets.  
    Added try/except blocks to handle errors and re-prompt until valid input.
2. **Unclear Data Organization** → Using plain tuples made it hard to manage movies.  
   Created a Movie class for a clean and structured data format.
3. **Average Calculation Error** → Division by zero occurred if no movies existed.  
   Checked if the movie list is empty before calculating the average.

Benefits

This project is beneficial in multiple ways:

* **Educational Use** → Helps students understand **OOP, functions, and exception handling** in Python.
* **Entertainment Industry** → Useful for **film budget comparisons** and financial analysis.
* **Research & Analysis** → Can be extended to include box office revenue for **profit analysis**.
* **Data Handling Skills** → Teaches how to manage, analyze, and report structured data effectively.

**Conclusion**

Movie Budget Analysis Project is a simple yet impactful Python program that combines good coding practices with practical outcomes. It successfully calculates averages, detects high-budget movies, and generates detailed reports. With future enhancements like graphs, profit/loss calculation, or database integration.

