Since I have used the Python programming language to develop this game, here is an introduction to **Python.**

Introduction to PYTHON-

Python is a widely-used general-purpose, high-level programming language. It was *created* by **Guido van Rossum** in 1991 and further developed by the Python Software Foundation. It was designed with an emphasis on code readability, and its syntax allows programmers to express their concepts in fewer lines of code.
There are two major Python versions: **Python 2** and **Python 3**.
Fact: I used Python3.10 in this project.

Developing any Application IDEs are proved to be very useful. I used VS Code as IDE here.

VS Code-

Visual Studio Code is a lightweight but powerful source code editor which runs on your desktop and is available for Windows, macOS, and Linux. It has built-in support for JavaScript, TypeScript, and Node.js and has a rich ecosystem of extensions for other languages (such as C++, C#, Java, Python, PHP, and Go).
Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git.

Why python is an efficient programming language -

1. Python's straightforward syntax is something that makes it easy to code in python.
2. Inherently, Python programs are text files containing instructions for the interpreter and are written in a text editor or IDE.
3. Writing an application in python is less time-consuming due to its short and simple syntax and the vast collection of built-in functions, modules, and libraries.
4. Python being fully object-oriented programming helps manage data and objects throughout the application.
5. Built-in data structures like list, dictionary, set, and tuple help in storing and working with data easily.
6. Python code is portable, so python code can be written on any platform and used on any.

About the game-

This game is made for those who want to practice arithmetic problems and become drastically good in arithmetic.

In this game, the player will have the option to choose any of the basic arithmetic operations from the menu given by entering the numeric code of the operation instance.
Then a new instance of the chosen operation will start, and you will get questions randomly generated automatically by the machine. In starting, questions will be easy, but as you keep

playing in the same instance of the game and your score grows, questions will start getting difficult.
Each question will somehow be connected to the instance you have chosen and the value of each symbol used in the question will also be displayed to you after the question. You will have to write your answer and then the program will evaluate the answer and scores you on the basis of your answer

Scoring process -

Each correct answer will be scored 4 and each wrong answer will deduct 2 from the score.

About the code of game-

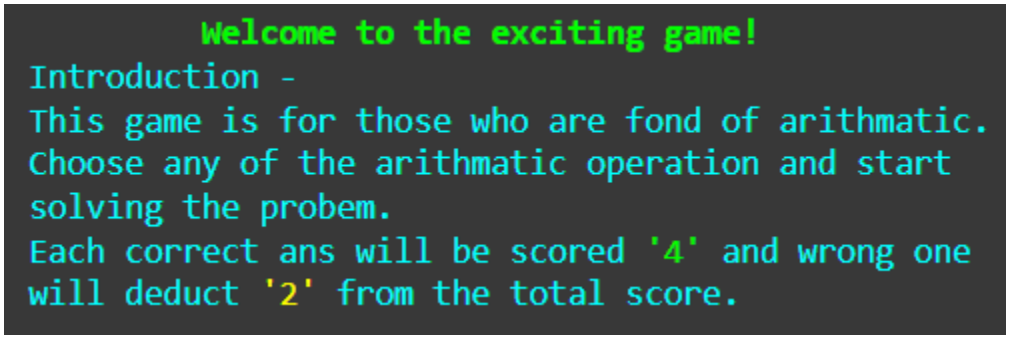
Outline of the program-

Functions

1. header
2. options
3. clear
4. generate_random_question
    a. recursive_generation
        i. returns recursively generated expression(question)
    b. returns question generated, values dictionary
5. start_new_instance
6. show_score_card
7. main

Working of the functions list above -

header - This function is made to print the welcome message along with the basic rules of the game.



options - this function shows the menu of all the operations accepted by the program.

```
Choose any of the options given below:
1. Addition
2. Subtraction
3. Multiplication
4. Float Division
5. Floor division
6. Modulos
7. Exponential
8. Randomly Generated
9. Show introduction.
10. Show score card.
11. Quit
```

clear - This function clears the terminal screen. This function works on all the terminals but doesn't work on services like google colab.

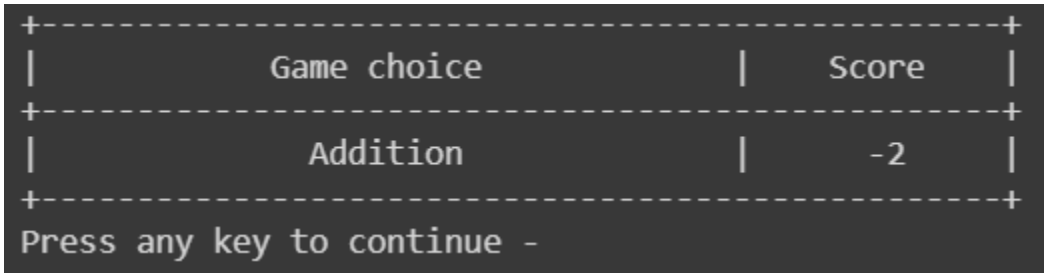
generate_random_question - This function is meant to handle the generating of random questions for the game play irrespective of the instance. This function needs to be nourished with these arguments - symbols, instance_type, num_array, level.
This function used another function defined and used inside of this named as recursive_generation.

> **recursive_generation -** recursive_generation function also required these arguments -symbols, symbols_used, op_dict, num_array, instance_type, count_left. This is a recursive function that chooses the symbols, their values and the defined operation based on the instance type provided

start_new_instance - This function starts the instance of the game chosen by the player(student) and runs an infinite loop that only lasts when the user enters .quit in the input area. Inside this loop randomly generated questions are displayed to the user and then the program waits for the user's answer and then compares the answer and then scores the user accordingly and also shows the message of answer being correct or wrong. In case of a wrong answer, the correct answer is also displayed to the user.

```
Starting new game instance...
Instance started!
Type .quit at the place of answer to quit the game.
Give float answers by rounding of to '2 decimal places'
+-------------------------------------------------+
|Question                                Score - 0
|a + x
|where: a = 6723, x = 9687
>>> 342342
Wrong answer    Answer: 16410.0
+-------------------------------------------------+
|Question                               Score - -2
|x + y
|where: y = 4509, x = 6066
>>> .quit
```

show_score_card -



main - This is acting as the triggering function, that by calling this function entire game play will start. This runs a while to keep playing till the user ends the game.