INFORMATICS INSTITUTE OF TECHNOLOGY

Informatics Institute of Technology

Foundation Certificate Programme

Module: Introduction to Programming in Python – P2

Module Leader: Dr. Damitha Karunaratne

Date of Submission: 10.04.2020

Assignment Type: Individual

Project Title:" Mathmania" Python based Math Game

Student Name: Yasiru Tharunda Jayatissa

Student ID: 2019103

Executive Summary

This report brings attention of the reader about a mathematical game called “MathMania” which is based on basic mathematical calculations. This report includes information about how this system was proposed, what are the characteristics of the system, specialty of the system comparing to others, etc. And also this report provides screenshots of the outputs of the system in different stages. This system was made to make it easy for mostly the children when it comes to Mathematics and mathematical calculations. This system was made with the hope that this will actively involve users in Mathematics than before. And even the source code of the system is also included in the report for the readers to get an idea about the sequence of the python code.

Acknowledgements

I am grateful because I managed to complete the group coursework regarding developing a Math game based on Python. Within the time given by our lecturer Dr. Damitha Karunaratne who is the module leader for Introduction to Programming in Python – P2. I also sincerely thank our lecturer for the tutorials Mr.Nishan Saliya Harnkahawa for the guidance and encouragement in finishing this report and also for teaching us. Last but not least, I would like to express my gratitude to my parents, friends, and respondents for the support they gave us.

Table of Content

1. Introduction and Description of the Project 1

2. Solution Outline 3

2.1 Game Menu 3

2.2 Quick game 3

2.3 Custom Game 6

2.3.1 Easy Mode 6

2.3.2 Medium Mode 8

2.3.3 Hard Mode 11

2.4 Past Game Details 14

2.5 Exit 16

3. The Sequence of The Code For Mathmania Math Game 17

4. Python Program For MathMania 18

4.1 Initialized and Created Variables 18

4.2 Display the Name of The Game 18

4.3 Calling The Menu Function And Asking User To Enter His/her Option 19

4.4 Exit option 19

4.5 Quick game Option 21

4.6 Custom Game Option 25

4.7 Past Game Details Option 39

4.8 **No option 46

References 48

List of figures

Figure 1--Displaying the name of the game 3
Figure 2--Game Menu..... 3
Figure 3--Welcome message displayed for the user if he/she chooses quick game option..... 3
Figure 4--Asking user to input his/her name 4
Figure 5--Displaying questions for quick game..... 4
Figure 6--Welcome message that gets shown before game results get printed 5
Figure 7--Game results for quick games..... 5
Figure 8--Welcome message displayed for the user if he/she chooses custom game option 6
Figure 9--Getting necessary inputs for easy mode from the user 6
Figure 10--Displaying questions in easy mode..... 7
Figure 11--Game results for easy mode..... 7
Figure 12--User entered data getting updated in the database..... 8
Figure 13--Getting necessary inputs for medium mode from the user 8
Figure 14--Displaying questions in the medium mode..... 9
Figure 15--Game results for medium mode..... 10
Figure 16--Gameplay information getting updated in a database after each gameplay 10
Figure 17--Getting necessary inputs for hard mode from the user 11
Figure 18--Displaying questions in the hard mode..... 12
Figure 19--Game results for hard mode..... 13
Figure 20--Database getting updated after each gameplay..... 13
Figure 21--Showing a welcome message to user after choosing the Past game details option 14
Figure 22--Displaying data about past gameplays to user 14
Figure 23--Getting necessary inputs from user and showing a welcome message..... 15
Figure 24--Displaying past gameplays related to each difficulty level according to user input... 15
Figure 25--User confirms to leave the system 16
Figure 26--User choosing to stay in the system even after choosing the option "exit" 16
Figure 27--User enters invalid command/option 16
Figure 28--The code sequence 17

1. Introduction and Description of the Project

In this assignment it was given to develop a simple mathematics game based on python. The python program should be developed giving user four options when playing the game.

1. Quick game
2. Custom game
3. Past game details
4. Exit

If the user chooses **"Quick game"** the program should allow the users to answer 5 simple questions based on addition. The operands has to be generated randomly and the operator has to be "+".

If the user chooses **"Custom game"** the program will offer user three different difficulties to choose before proceeding further. The program has to be developed giving user the option to choose number of questions he/she wants to answer.

I. Easy mode – If the user chooses the option easy game as in the quick game questions need to be generated based on addition. The operands has to be random and within the range 0 to 10.
II. Medium mode – If the user chooses medium mode questions need to be generated based on addition and subtraction. The operands need to be randomly generated and they have to be within 0 to 50 range.
III. Hard mode – If the user chooses hard mode questions need to be generated based on addition, subtraction and multiplication. The operands need to be randomly generated and they have to be within 0 to 100 range.

And before playing both game modes the program should obtain user's name, number of questions he/she wants to answer, what mode they want to play and with these information Number of correct answers given and final score(percentage) of the user has also needs to be stored.

After each and every game play in any of the game modes game results needs to be shown to the user. This has to include user's name, the mode he/she has played with, number of questions he/she had answered, and the questions has to be displayed the exact same way as they were generated with the answer provided by user, the correct answer and whether the user's answer is correct or not.

As an example 5 + 2 =8 (Answer 7) [Incorrect]

If the user chooses the option **"Past game details"**, results of the past plays needs to be displayed with the names of the users, how many questions were generated, how many correct answers were given and with the score they obtained. For this the pre-stored information which was taken before playing has to be used. And also the number of gameplays which was played so far also needs to be displayed.

Name	correct	Total Questions	Score
John	5	10	50%

Total numbers of game plays are 25.

And when the user chooses the option **"Exit"** the program needs to stop running.

2. Solution Outline

In the very beginning of the program I used multiline print statements and symbols to display the name of the math game to the users.

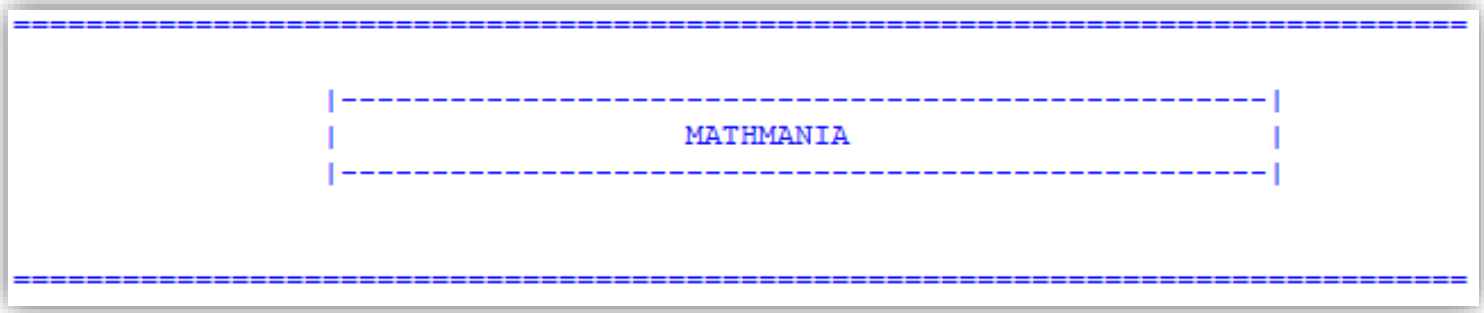

Figure 1--Displaying the name of the game

2.1 Game Menu

First to create the game menu I used usual print statements that are used in python. Then I used an input statement to get any of the options among them and assigned it to a variable.

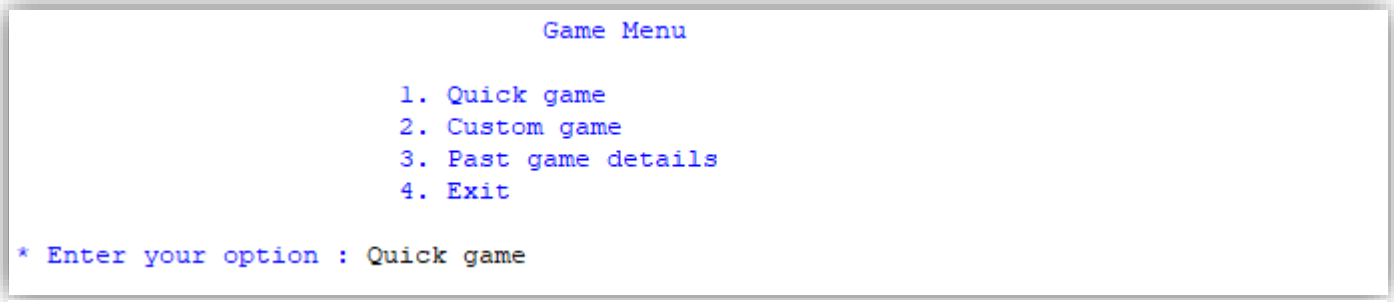

Figure 2--Game Menu

2.2 Quick game

First I developed the program to show the user a welcome message when selecting quick game option by using multiline print statements.

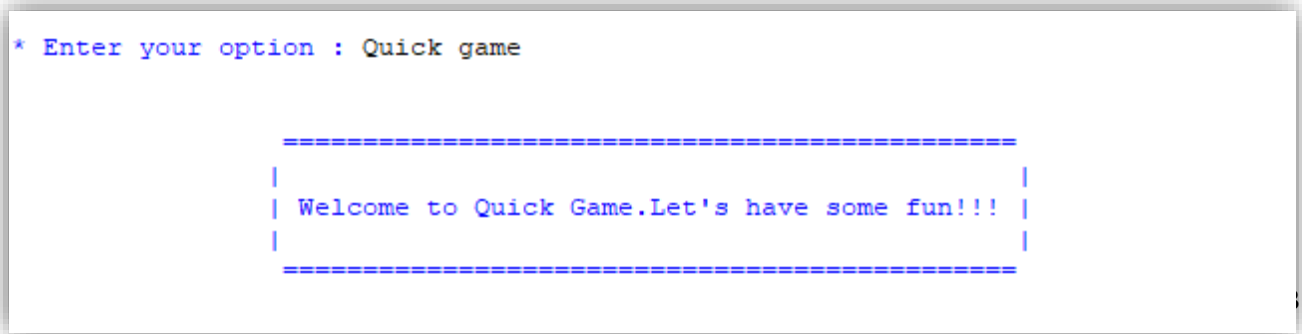

Figure 3--Welcome message displayed for the user if he/she chooses quick game option

At first I developed the quick game program. If the users choose quick game option his/her name it will be taken using an input statement and get assigned to a variable called “name”.

```
                    Game Play

* Enter your name :
```

Figure 4--Asking user to input his/her name

Variables used to develop quick game and easy mode in custom game play:

```
queslist = [] # List that contains generated questions
canslist = [] # List that contains correct answers
uanslist = [] # List that contains users' answers
count = 0
correctcount = 0
score = 0
```

To generate random operands I imported the “random module” and used “randrange” attribute. And then I used f-strings in python to type the format of the questions that will be generated and made a list (queslist) to assign all those questions in an orderly manner to that created list. Then using print statements I created a way to print all those questions. And then I used an input statement to get user’s answer and assigned that answer into a variable. In here I created an if statement to find out whether that variable is empty or not. If that variable is not empty it will add a one to the “count” variable and at the results page it will display to user for how many questions he/she has answered. And this user’s answer is also getting added to another list (uanslist). Then I created another variable to calculate the correct answer for the question and that variable values will also get stored in another list. To add values for each and every questions and to continue generating and printing questions I used a for loop.

```
          Answer these questions!!!
  8 + 1 ?
  9 + 10 ?
  9 + 6 ?
  0 + 2 ?
  8 + 4 ?
```

Figure 5--Displaying questions for quick game

Displaying Results:

Before showing the results I used multiline print statements again to show user a welcome message when entering into the results page.

```
                        Game results

**************************************************************
|                          WELLDONE!!!                       |
|                                                            |
|              Let's see how well you have played            |
|                                                            |
|                                                            |
**************************************************************
```

Figure 6--Welcome message that gets shown before game results get printed

And then I used print statements to show the user he has inserted before using the variable "name". And then I programmed to print how many questions he/she has answered. I used an if statement to check whether the user's answer is equal to the correct answer and if that is correct it will add one to the variable "correctcount". Here I used a for loop to run the program 5 times to check all 5 questions which were generated. And then I used a print statement to print how many questions he/she has correctly answered. Then I calculated the score using calculations and printed the final score he/she has obtained.

```
* Your name is  Yasiru
* You answered  5 Questions

                6 + 7 = 13 (Answer13)[correct]
                6 + 8 = 5 (Answer14)[Incorrect]
                7 + 9 = 16 (Answer16)[correct]
                3 + 2 = 5 (Answer5)[correct]
                7 + 8 = 15 (Answer15)[correct]

* you have provided correct answers for  4 questions

* your score is  80.0
```

Figure 7--Game results for quick games

And to store gameplay information of the user I created a database using MySQL connector and made a connection to the localhost and stored all the variable information related to the variables like "name", "correctcount" and "score".

2.3 Custom Game

First I developed the program to show the user a welcome message when selecting custom game option by using multiline print statements.

```
* Enter your option : Custom game


        ============================================================
        |                                                          |
        | Welcome to Custom Game.Let's check your calculation power |
        |                                                          |
        ============================================================
```

Figure 8--Welcome message displayed for the user if he/she chooses custom game option

2.3.1 Easy Mode

If the user chooses Easy game mode his/her name will be taken using an input statement and get assigned to a variable called “name” and number of questions they want to answer will also be taken as an input and get assigned to another variable called “qneed”.

```
                                Game Play

* Select your difficulty level (Easy / Medium / Hard) : Easy

* Enter your name : Yasiru

* How many questions you need to answer : 12
```

Figure 9--Getting necessary inputs for easy mode from the user

To generate random operands I imported the “random module” and used “randrange” attribute. And then I used f-strings in python to type the format of the questions that will be generated and made a list (queslist) to assign all those questions in an orderly manner to that created list. Then using print statements I created a way to print all those questions. And then I used an input statement to get user’s answer and assigned that answer into a variable. In here I created an if statement to find out whether that variable is empty or not. If that variable is not empty it will add a one to the “count” variable and at the results page it will display to user for how many questions he/she has answered. And this user’s answer is also getting added to another list

(uanslist). Then I created another variable to calculate the correct answer for the question and that variable values will also get stored in another list. To add values for each and every questions and to continue generating and printing questions I used a for loop.

```
                    Think Well And Answer these questions!!!
   9 + 8 ?
   8 + 1 ?
   3 + 1 ?
   2 + 7 ?
   9 + 0 ?
   10 + 7 ?
   7 + 0 ?
   3 + 9 ?
```

Figure 10--Displaying questions in easy mode

Displaying Results:

In here also the same welcome message was shown after the gameplay.

And then I used print statements to show the user he has inserted before using the variable "name". I used a print statements to show the user which game mode he/she has played. And then I programmed to print how many questions were generated and among them how many questions he/she has answered. I used an if statement to check whether the user's answer is equal to the correct answer and if that is correct it will add one to the variable "correctcount". Here I used a for loop to run the program until it reaches the "qneed" variable value. And then I used a print statement to print how many questions he/she has correctly answered. Then I calculated the score using calculations and printed the final score he/she has obtained.

```
* Your name is  Yasiru

* You played the game with Easy mode

* There were  6 Questions generated

* You answered  6 Questions

                1 + 0 = 1 (Answer 1) [correct]
                10 + 9 = 5 (Answer 19) [Incorrect]
                3 + 9 = 6 (Answer 12) [Incorrect]
                4 + 7 = 2 (Answer 11) [Incorrect]
                0 + 2 = 2 (Answer 2) [correct]
                4 + 9 = 13 (Answer 13) [correct]

* you have provided correct answers for  3 questions

* your score is  50.0 %
```

Figure 11--Game results for easy mode

And to store gameplay information of the user I created a database using MySQL connector and made a connection to the localhost and stored all the variable information related to the variables like ”name”, ”correctcount”, ”qneed ” and “score”. All information inserted by the user will get saved in the database under a specific table as follows.

Name	Total_Questions	Correct_Answers	Score
Yasiru	5	5	100

Figure 12--User entered data getting updated in the database

2.3.2 Medium Mode

If the user chooses Medium game mode his/her name will be taken using an input statement and get assigned to a variable called “name” and number of questions they want to answer will also be taken as an input and get assigned to another variable called “qneed”.

```
* Select your difficulty level (Easy / Medium / Hard) : Medium

* Enter your name : Yasiru

* How many questions you need to answer : 6
```

Figure 13--Getting necessary inputs for medium mode from the user

Variables used to develop medium mode in custom game play:

```
queslist = [] # List that contains generated questions
canslist = [] # List that contains correct answers
uanslist = [] # List that contains users' answers
m_operators = ["+","-"]
count = 0
correctcount = 0
```

To generate random operands and operators I imported the “random module” and used “randrange” and “choice” attributes respectively. And then I used f-strings in python to type the format of the questions that will be generated and made a list (queslist) to assign all those questions in an orderly manner to that created list. Then using print statements I created a way to print all those questions. And then I used an input statement to get user’s answer and assigned that answer into a variable. In here I created an if statement to find out whether that variable is empty or not. If that variable is not empty it will add a one to the “count” variable and at the results page it will display to user for how many questions he/she has answered. And this user’s answer is also getting added to another list (uanslist). Then I created another variable to calculate the correct answer for the question and that variable values will also get stored in another list. To add values for each and every questions and to continue generating and printing questions I used a for loop.

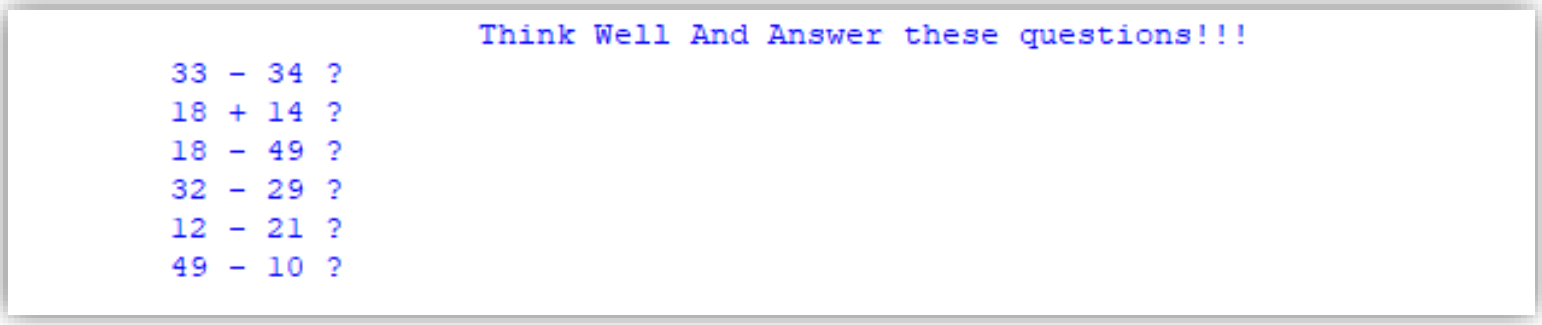

Figure 14--Displaying questions in the medium mode

Displaying Results:

In here also the same welcome message was shown after the gameplay.

And then I used print statements to show the user he has inserted before using the variable “name”. I used a print statements to show the user which game mode he/she has played. And then I programmed to print how many questions were generated and among them how many questions he/she has answered. I used an if statement to check whether the user’s answer is equal to the correct answer and if that is correct it will add one to the variable “correctcount”. Here I used a for loop to run the program until it reaches the “qneed” variable value. And then I used a print statement to print how many questions he/she has correctly answered. Then I calculated the score using calculations and printed the final score he/she has obtained

```
* Your name is  Yasiru

* You played the game with Medium mode

* There were  8 Questions generated

* You answered  8 Questions

                0 + 30 = 30 (Answer 30) [correct]
                44 + 45 = 89 (Answer 89) [correct]
                25 + 30 = 55 (Answer 55) [correct]
                26 + 8 = 4 (Answer 34) [Incorrect]
                16 + 27 = 13 (Answer 43) [Incorrect]
                32 + 23 = 55 (Answer 55) [correct]
                20 + 38 = 58 (Answer 58) [correct]
                24 + 46 = 70 (Answer 70) [correct]

* you have provided correct answers for  6 questions

* your score is  75.0 %
```

Figure 15--Game results for medium mode

And to store gameplay information of the user I created a database using MySQL connector and made a connection to the localhost and stored all the variable information related to the variables like ”name”, ”correctcount” , ”qneed ” and “score”. All information inserted by the user will get saved in the database under a specific table as follows.

Name	Total_Questions	Correct_Answers	Score
Yasiru	5	3	60.
Kushan	7	6	85.

Figure 16--Gameplay information getting updated in a database after each gameplay

2.3.3 Hard Mode

If the user chooses Medium game mode his/her name will be taken using an input statement and get assigned to a variable called “name” and number of questions they want to answer will also be taken as an input and get assigned to another variable called “qneed”.

```
* Select your difficulty level (Easy / Medium / Hard) : Hard

* Enter your name : Yasiru

* How many questions you need to answer : 9
```

Figure 17--Getting necessary inputs for hard mode from the user

Variables used to develop medium mode in custom game play:

```
queslist = [] # List that contains generated questions
canslist = [] # List that contains correct answers
uanslist = [] # List that contains users' answers
h_operators = ["+","-","*"]
count = 0
correctcount = 0
```

To generate random operands and operators I imported the “random module” and used “randrange” and “choice” attributes respectively. And then I used f-strings in python to type the format of the questions that will be generated and made a list (queslist) to assign all those questions in an orderly manner to that created list. Then using print statements I created a way to print all those questions. And then I used an input statement to get user’s answer and assigned that answer into a variable. In here I created an if statement to find out whether that variable is

empty or not. If that variable is not empty it will add a one to the “count” variable and at the results page it will display to user for how many questions he/she has answered. And this user’s answer is also getting added to another list (uanslist). Then I created another variable to calculate the correct answer for the question and that variable values will also get stored in another list. To add values for each and every questions and to continue generating and printing questions I used a for loop.

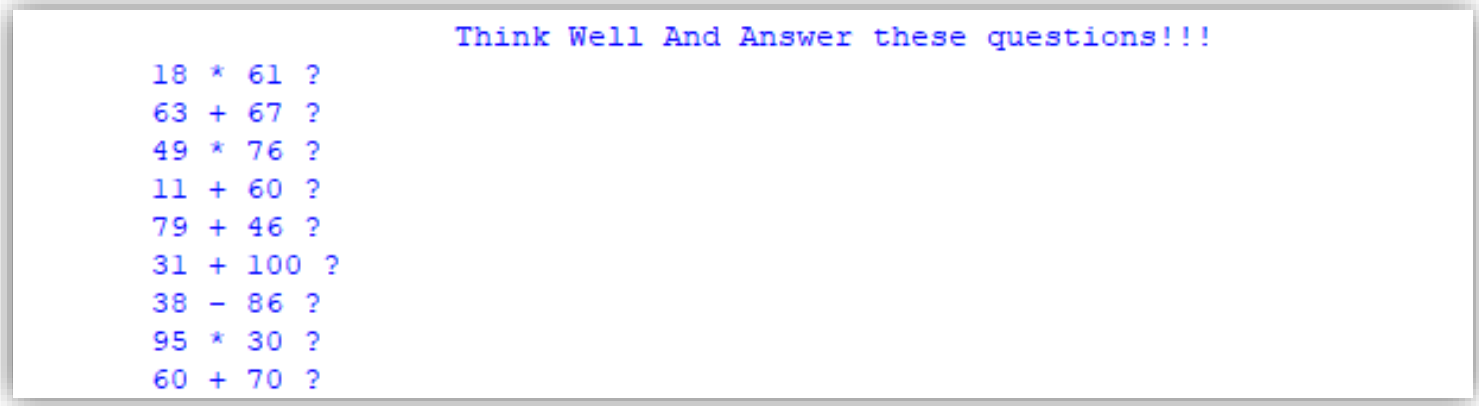

Figure 18--Displaying questions in the hard mode

Displaying Results:

In here also the same welcome message was shown after the gameplay.

And then I used print statements to show the user he has inserted before using the variable “name”. I used a print statements to show the user which game mode he/she has played. And then I programmed to print how many questions were generated and among them how many questions he/she has answered. I used an if statement to check whether the user’s answer is equal to the correct answer and if that is correct it will add one to the variable “correctcount”. Here I used a for loop to run the program until it reaches the “qneed” variable value. And then I used a print statement to print how many questions he/she has correctly answered. Then I calculated the score using calculations and printed the final score he/she has obtained.

```
* Your name is  Yasiru

* You played the game with Hard mode

* There were  7 Questions generated

* You answered  7 Questions

                1 + 62 = 63 (Answer 63)[correct]
                58 * 65 = 25 (Answer 3770)[Incorrect]
                59 * 2 = 118 (Answer 118)[correct]
                48 + 13 = 61 (Answer 61)[correct]
                59 + 16 = 75 (Answer 75)[correct]
                47 * 51 = 5 (Answer 2397)[Incorrect]
                44 + 62 = 106 (Answer 106)[correct]

* you have provided correct answers for  5 questions

* your score is  71.42857142857143 %
```

Figure 19--Game results for hard mode

And to store gameplay information of the user I created a database using MySQL connector and made a connection to the localhost and stored all the variable information related to the variables like ”name”, ”correctcount” , ”qneed ” and “score”. All information inserted by the user will get saved in the database under a specific table as follows.

Name	Total_Questions	Correct_Answers	Score
Yasiru	8	5	62.
Kushan	6	5	83.

Figure 20--Database getting updated after each gameplay

2.4 Past Game Details

If the user chooses the option past game details, I programmed the system to ask a question from the user “what is the game type?”, and then I used if statements to differentiate what will happen if user chooses quick game and custom game. If the user chooses quick game, a welcome message will be shown to the user which is programmed using multiline print statements.

```
* Enter your option : Past game details

* What is the game type( Quick game / Custom game ) : Quick game


                 ================================================
                |                                                |
                |        Welcome to past game details!!!         |
                |                                                |
                 ================================================
```

Figure 21--Showing a welcome message to user after choosing the Past game details option

And then I made a connection with the database I made before to store data, using MySQL connector. Then I used queries to extract necessary information from the database and stored them in different variables. Then by using prettytable module I created a table to display to the user and named the column as Name, Total questions, Correct answers and score. And then using those pre-stored values in the variables I added columns including the variables.

```
                                Game Play
+-------------+-----------------+-----------------+----------+
|     Name    | Total Questions | Correct Answers |  Score   |
+-------------+-----------------+-----------------+----------+
|  ('Yasir',) |      ('5',)     |      ('3',)     | ('60.',) |
| ('Kushan',) |      ('5',)     |      ('5',)     | ('100',) |
+-------------+-----------------+-----------------+----------+
```

Figure 22--Displaying data about past gameplays to user

If the user chooses the option past game details, I programmed the system to ask a question from the user “what is the game type?”, and then I used if statements to differentiate what will happen if user chooses quick game and custom game. If the user chooses custom game, I programmed the system to ask a question from the user about what information related to which game mode needs to be displayed. And then a welcome message will be displayed to the user using multiline print statements.

```
* Enter your option : Past game details

* What is the game type( Quick game / Custom game ) : Custom game

* What is the game mode (Easy / Medium / Hard) :Hard


 ================================================
|                                                |
|        Welcome to Past Game Details!!!         |
|                                                |
 ================================================
```

Figure 23--Getting necessary inputs from user and showing a welcome message

And then I made a connection with the database I made before to store data, using MySQL connector. Then I used queries to extract necessary information from the database and stored them in different variables. Then by using prettytable module I created a table to display to the user and named the column as Name, Total questions, Correct answers and score. And then using those pre-stored values in the variables I added columns including the variables. And in here for all three difficlty levels (Easy, Medium, Hard) same code was used.

```
                                Game Play
+-------------+-----------------+-----------------+----------+
|     Name    | Total Questions | Correct Answers |  Score   |
+-------------+-----------------+-----------------+----------+
| ('Yasiru',) |      ('6',)     |      ('6',)     | ('100',) |
| ('Kushan',) |      ('8',)     |      ('8',)     | ('100',) |
+-------------+-----------------+-----------------+----------+
```

Figure 24--Displaying past gameplays related to each difficulty level according to user input

2.5 Exit

I programmed the code to execute as a repetitive loop using for loops until the user selects the option “exit”. If the users choose the option “exit” a question will be asked using an input statements to make sure whether the user really wants to exit the system or not. And if the users select the option “yes” a message will be shown to the user as follows and the program will stop executing. And the user will have to re-run the program again.

```
* Enter your option : Exit

* Are you sure You want to exit (Y/N) : Y


                ==================================================
               |                                                  |
               | Good Bye!!!. Come Again When you are ready to play |
               |                                                  |
                ==================================================


>>>
```

Figure 25--User confirms to leave the system

And if the users choose the option “no” program will continue executing.

```
* Enter your option : Exit

* Are you sure You want to exit (Y/N) : N
================================================================================
                                  Game Menu

                          1. Quick game
                          2. Custom game
                          3. Past game details
                          4. Exit

* Enter your option :
```

Figure 26--User choosing to stay in the system even after choosing the option "exit"

If the user leaves the option bar without choosing any option or enter any incorrect option I programmed the system to show a message as follows using multiline print statements.

```
* Enter your option :


        ======================================================================
        |                                                                    |
        |               follow the instructions carefully!!!                 |
        |                                                                    |
        ======================================================================
```

Figure 27--User enters invalid command/option

3. The Sequence of The Code For Mathmania Math Game

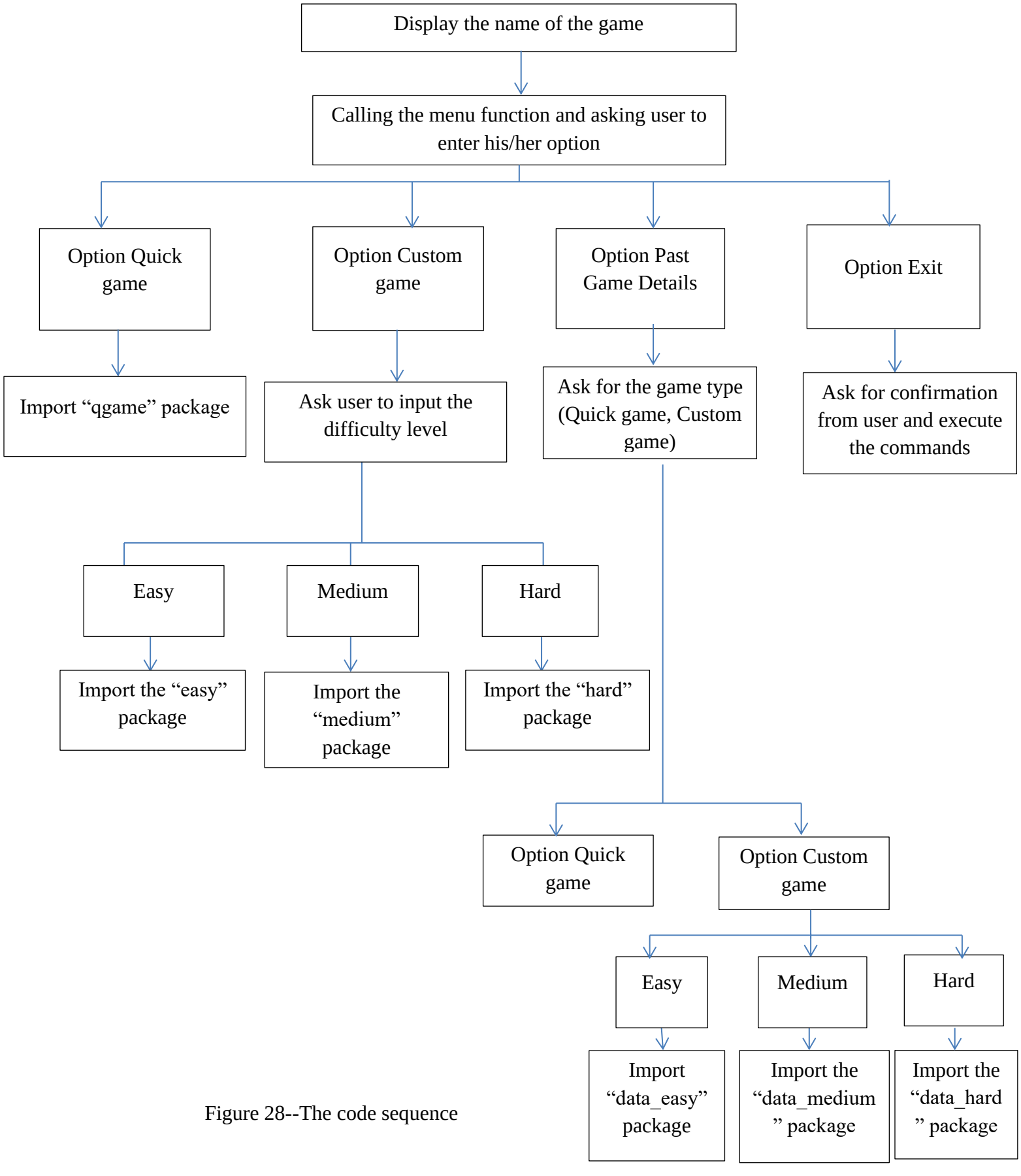

Figure 28--The code sequence

4. Python Program For MathMania

4.1 Initialized and Created Variables

```
# List variable declaration

queslist = [] # List that contains generated questions

canslist = [] # List that contains correct answers

uanslist = [] # List that contains users' answers

m_operators = ["+","-"]

h_operators = ["+","-","*"]

# creating and initializing variables

count = 0

name = 0

dflevel = 0

qneed = 0

uans = 0

correctcount = 0
```

4.2 Display the Name of The Game

```
# Basic game menu

print
("============================================================")
```

```
print ('''
            |-----------------------------------------------|
            |                MATHMANIA                      |
            |-----------------------------------------------|
     ''')
print
("=================================================================")
```

4.3 Calling The Menu Function And Asking User To Enter His/her Option

```
for i in range (5):
    from menu import game_menu
    game_menu ()
    # Showing user the message to input his/her option
    option = input ("\t\n* Enter your option : ")
```

Game Menu

```
# The basic game menu of math mania
def game_menu():
    print("\t\t\t\t Game Menu")
    print("\n\t\t\t1. Quick game")
    print("\t\t\t2. Custom game")
    print("\t\t\t3. Past game details")
    print("\t\t\t4. Exit")
```

4.4 Exit option

```
# User choosing the option Exit
    if (option == "Exit"):
        eop = input("\n* Are you sure You want to exit (Y/N) : ") #eop stands for exit option
        if eop == "Y":


  print ('''
              ==================================================
             |                                                  |
             |        Good Bye!!!. Come Again When you are ready to play          |
             |                                                  |
              ==================================================
            ''')
            break
       print("============================================================")
```

4.5 Quick game Option

```
 # User choosing the option quick game
    elif (option == "Quick game"):
        # Calling the q_quick function which is in qgame module in quickgame package
        import quickgame.qgame
        quickgame.qgame.q_quick()
         print("============================================================")
```

qgame module

```
# This is what's going to happen if the user chooses Quick game option
def q_quick():
    # A Welcome message to the user
    print('''
              ===============================================
             |                                                 |
             |          Welcome to Quick Game.Let's have some fun!!!           |
             |                                                 |
              ===============================================
    ''')
    print("\t\t\t\t Game Play")
```

```
# Calling variables and lists that are needed for the functioning of this module
    queslist = [] # List that contains generated questions
    canslist = [] # List that contains correct answers
    uanslist = [] # List that contains users' answers
    count = 0
    correctcount = 0
    score = 0
     # Asking user to input his/her name
    name = input("\n* Enter your name :")
    # Giving user 5 questions to answer using while loops
    print ("\n\t\t\t Answer these questions!!!")
    for x in range (5):
        import random
        rand1 = random.randrange(0,11)
        rand2 = random.randrange(0,11)
        ques = (f'{rand1} + {rand2} = ' ) # Creating question as a variable using f strings
        queslist.append(ques) # Input the generated questions to the list
        print("\t",rand1 , "+" , rand2, "?",end=" ")
        uans = input()
        if uans != "" :
            count + = 1
```

```
  else:
      pass
   uanslist.append(uans) # Input the users' answer into the list
   cans = str(rand1 + rand2)
   canslist.append(cans) # Input the correct answer into the list
# Displaying game results
print("\n\t\t\t\t Game results ")
print('''
    ************************************************************************
    |                                WELLDONE!!!                           |
    |                                                                      |
    |                        Let's see how well you have played            |
    |                                                                      |
    |                                                                      |
    ************************************************************************
   ''')
# Showing user the data like his name, difficulty level and no.of questions he/she answered
print("* Your name is ",name)
print("* You answered ",count, "Questions\n")
for y in range (5):
   if canslist[y] == uanslist[y]:
```

```
        correctcount += 1
        print("\t\t" f' {queslist[y]}{uanslist[y]} (Answer{canslist[y]})[correct]')# Printing the
            output with the created lists
else:
        print("\t\t" f' {queslist[y]}{uanslist[y]} (Answer{canslist[y]})[Incorrect]' )
print("\n* you have provided correct answers for ", correctcount, "questions")
score = correctcount/5 * 100
print("\n* your score is ", score )
# Establishing connection with the database
import mysql.connector
db = mysql.connector.connect(user="Useradmin",password="user123456",
                                        host="127.0.0.1" ,database="game")
cursor = db.cursor ()
sqltext = "INSERT INTO Quick_game (Name, Total_Questions, Correct_Answers, Score )
                    VALUES(%s,%s,%s,%s)"
myvalues = (name, 5, correctcount, score )
cursor.execute (sqltext, myvalues)
myvalues = (name, 5, correctcount, score)
cursor.execute (sqltext, myvalues)
db.commit()
db.close()
```

4.6 Custom Game Option

```
# User choosing the option Custom game

elif (option == "Custom game"):

    import customgame.welcome

    customgame.welcome.custom_welcome()

    dflevel = input("\n* Select your difficulty level (Easy / Medium / Hard) : ")# dflevel stands
                    for difficlty level

    if dflevel == "Easy":

       import customgame.easy

       customgame.easy.easy_mode()
       print("============================================================")

    elif dflevel == "Medium":

       import customgame.medium

       customgame.medium.medium_mode()
       print("============================================================")

    elif dflevel == "Hard":

       import customgame.hard

       customgame.hard.hard_mode()
       print("============================================================")
```

easy_mode module

```
def easy_mode ():

#Easy mode

# Taking the user's name, difficlty level and number of questions he/she wants to answer as an
  input to enter to the database when saving

name = input("\n* Enter your name : " )

qneed = int(input("\n* How many questions you need to answer : ")) # qneed stands for
                        questions need to answer

# Calling variables and lists that are needed for the functioning of this module

queslist = []  # List that contains generated questions

canslist = [] # List that contains correct answers

uanslist = [] # List that contains users' answers

m_operators = ["+","-"]

h_operators = ["+","-","*"]

count = 0

correctcount = 0


# Giving user questions according to his inputs

print ("\n\t\t\t Think Well And Answer these questions!!!")

for x in range (qneed):

      import random
```

```
rand1 = random.randrange(0,11)

rand2 = random.randrange(0,11)

ques = (f'{rand1} + {rand2} = ' ) # Creating question as a variable using f strings

queslist.append(ques) # Input the generated questions to the list

print("\t",rand1 , "+" , rand2, "?",end=" ")

uans = input()

if uans != "" : #  If users' answer is not empty  then count = count+1

    count + = 1

else:

    pass

uanslist.append(uans)# Input the users' answer into the list

cans = str(rand1 + rand2)

canslist.append (cans) # Input the correct answer into the list
```

```
# Displaying game results
print ("\n\t\t\t\t Game results ")
print('''
        ************************************************************
        |                          WELLDONE!!!                     |
        |                                                          |
        |                  Let's see how well you have played      |
        |                                                          |
        |                                                          |
        ************************************************************
      ''')
# Showing user the data like his name, difficulty level and no.of questions he/she answered
print("* Your name is ",name, "\n")
print("* You played the game with Easy mode\n")
print("* There were ",qneed, "Questions generated\n")
print("* You answered ",count, "Questions\n")
for y in range (qneed):
    if canslist[y] == uanslist[y]:
        correctcount += 1 # If the answer is correct correctcount = correctcount + 1
        print("\t\t" f' {queslist[y]}{uanslist[ y ]} (Answer {canslist[y]}) [correct]')# Printing the
            output with the created lists
```

```
    else:
        print("\t\t" f' {queslist[y]}{uanslist[ y ]} (Answer {canslist[y]}) [Incorrect]' )
print("\n* you have provided correct answers for ", correctcount, "questions")
score = correctcount/qneed * 100
print("\n* your score is ", score ,"%") #  Calculating the perecentage
# Establishing connection with the database
import mysql.connector
db = mysql.connector.connect(user="Useradmin",password="user123456",
                                        host="127.0.0.1",database="customgame")
cursor = db.cursor()
sqltext = "INSERT INTO Easy_mode (Name, Total_Questions, Correct_Answers, Score)
          VALUES (%s,%s,%s,%s)"
myvalues = (name,qneed,correctcount,score)
cursor.execute(sqltext,myvalues)
db.commit()
db.close()
```

medium_mode module

```
def medium_mode ():

# Medium Mode

# Taking the user's name, difficlty level and number of questions he/she wants to answer as an input to enter to the database when saving

name = input("\n* Enter your name : " )

qneed = int(input("\n* How many questions you need to answer : ")) # qneed stands for
        questions need to answer

# Calling variables and lists that are needed for the functioning of this module

queslist = []  # List that contains generated questions

canslist = [] # List that contains correct answers

uanslist = [] # List that contains users' answers

m_operators = ["+","-"]

h_operators = ["+","-","*"]

count = 0

correctcount = 0


# Giving user questions according to his inputs

print("\n\t\t\t Think Well And Answer these questions!!!")

for x in range (qneed):

    import random
```

```
rand1 = random.randrange(0,51)

rand2 = random.randrange(0,51)

rand3 = random.choice(m_operators) #  In here i am going to use random.choice

if(rand3 == "+"):

    ques = (f'{rand1} + {rand2} = ') # Creating question as a variable using f strings

    queslist.append(ques) # Input the generated questions to the list

    print("\t",rand1 , "+" , rand2, "?" ,end=" ")

    uans = input()

    if uans != "" :

        count += 1

    else:

        pass

    uanslist.append(uans) # Input the users' answer into the list

    cans = str(rand1 + rand2)

    canslist.append(cans) # Input the correct answer into the list

elif(rand3 == "-"):

    ques = (f'{rand1} - {rand2} = ') # Creating question as a variable using f strings

    queslist.append(ques) # Input the generated questions to the list

    print("\t",rand1 , "-" , rand2, "?" ,end=" ")

    uans = input()

    if uans != "" :
```

```
            count += 1
        else:
            pass
        uanslist.append(uans) # Input the users' answer into the list
        cans = str(rand1 - rand2)
        canslist.append(cans) # Input the correct answer into the list
# Displaying game results
print("\n\t\t\t\t Game results ")
print('''
    ************************************************************
    |                          WELLDONE!!!                     |
    |                                                          |
    |                  Let's see how well you have played      |
    |                                                          |
    |                                                          |
    ************************************************************
    ''')
# Showing user the data like his name, difficulty level and no.of questions he/she answered
print("* Your name is ", name,"\n")
print("* You played the game with Medium mode\n")
print("* There were ",qneed, "Questions generated\n")
```

```
print("* You answered ",count, "Questions\n")
for y in range (qneed):
    if canslist[y] == uanslist[y]:
        correctcount += 1
        print("\t\t" f'{queslist[y]}{uanslist[ y ]} (Answer {canslist[y]}) [correct]') # Printing the
            output with the created lists
    else:
        print("\t\t" f'{queslist[y]}{uanslist[ y ]} (Answer {canslist[y]}) [Incorrect]' )
print("\n* you have provided correct answers for ", correctcount, "questions")
score = correctcount/qneed * 100
print("\n* your score is ",score ,"%")
# Establishing connection with the database
import mysql.connector
db = mysql.connector.connect(user="Useradmin",password="user123456",
                             host="127.0.0.1",database="customgame")
cursor = db.cursor()
sqltext = "INSERT INTO Medium_mode
            (Name,Total_Questions,Correct_Answers,Score)VALUES (%s,%s,%s,%s)"
myvalues = (name, qneed, correctcount, score)
cursor.execute (sqltext, myvalues)
db.commit ()
db.close ()
```

hard_mode Module

```
def hard_mode ():

# Hard Mode

# Taking the user's name, difficlty level and number of questions he/she wants to answer as an
  input to enter to the database when saving

name = input("\n* Enter your name : " )

qneed = int(input("\n* How many questions you need to answer : "))# qneed stands for questions
need to answer

# Calling variables and lists that are needed for the functioning of this module

queslist = [] # List that contains generated questions

canslist = [] # List that contains correct answers

uanslist = [] # List that contains users' answers

m_operators = ["+","-"]

h_operators = ["+","-","*"]

count = 0

correctcount = 0

# Giving user questions according to his inputs

print ("\n\t\t\t Think Well And Answer these questions!!!")

for x in range(qneed):

    import random

    rand1 = random.randrange(0,101)
```

```
rand2 = random.randrange(0,101)
rand3 = random.choice(h_operators) # In here i am going to use random.choice
if(rand3 == "+"):
    ques = (f'{rand1} + {rand2} = ') # Creating question as a variable using f strings
    queslist.append(ques) # Input the generated questions to the list
    print("\t",rand1 , "+" , rand2, "?" ,end=" ")
    uans = input()
    if uans != "" :
        count += 1
    else:
        pass
    uanslist.append(uans) # Input the users' answer into the list
    cans = str(rand1 + rand2)
    canslist.append(cans) # Input the correct answer into the list
elif(rand3 == "-"):
    ques = (f'{rand1} - {rand2} = ') # Creating question as a variable using f strings
    queslist.append(ques) # Input the generated questions to the list
    print("\t",rand1 , "-" , rand2, "?" ,end=" ")
    uans = input()
    if uans != "" :
        count += 1
```

```
        else:
            pass
        uanslist.append(uans) # Input the users' answer into the list
        cans = str(rand1 - rand2)
        canslist.append(cans) # Input the correct answer into the list
    elif(rand3 == "*"):
        ques = (f'{rand1} * {rand2} = ') # Creating question as a variable using f strings
        queslist.append(ques) # Input the generated questions to the list
        print("\t",rand1 , "*" , rand2, "?" ,end=" ")
        uans = input()
        if uans != "" :
            count += 1
        else:
            pass
        uanslist.append(uans) # Input the users' answer into the list
        cans = str(rand1 * rand2)
        canslist.append(cans) # Input the correct answer into the list


# Displaying game results
print("\n\t\t\t\t Game results ")
```

```
print('''
        ************************************************************
        |                          WELLDONE!!!                     |
        |                                                          |
        |                  Let's see how well you have played      |
        |                                                          |
        |                                                          |
        ************************************************************
       ''')
# Showing user the data like his name, difficulty level and no.of questions he/she answered
print("* Your name is ", name,"\n")
print("* You played the game with Hard mode\n")
print("* There were ",qneed, "Questions generated\n")
print("* You answered ",count, "Questions\n")
for y in range(qneed):
    if canslist[y] == uanslist[y]:
        correctcount += 1
        print("\t\t" f'{queslist[y]}{uanslist[ y ]} (Answer {canslist[y]})[correct]') # Printing the
            output with the created lists
    else:
        print("\t\t" f'{queslist[y]}{uanslist[ y ]} (Answer {canslist[y]})[Incorrect]' )
```

```
print("\n* you have provided correct answers for ", correctcount, "questions")

score = correctcount/qneed * 100

print("\n* your score is ", score ,"%")

# Establishing connection with the database

import mysql.connector

db = mysql.connector.connect(user="Useradmin",password="user123456",

                             host="127.0.0.1",database="customgame")

cursor = db.cursor()

sqltext = "INSERT INTO Hard_mode (Name, Total_Questions, Correct_Answers, Score)
          VALUES (%s,%s,%s,%s)"

myvalues = (name,qneed,correctcount,score)

cursor.execute(sqltext,myvalues)

db.commit ()

db.close()
```

4.7 Past Game Details Option

```
elif (option == "Past game details"):
    fetchreq = input("\n* What is the game type( Quick game / Custom game ) : ")
    if fetchreq == "Quick game":
        import results.quickgame
        results.quickgame.data_qgame()
        print("=========================================================")
    elif fetchreq == "Custom game":
        mode = input("\n* What is the game mode (Easy / Medium / Hard) :")
        import results.customgame
        # A Welcome message to the user
        print('''
            ============================================
           |                                            |
           |          Welcome to Past Game Details!!!   |
           |                                            |
            ============================================
        ''')
        print("\t\t\t\t Game Play")
        if (mode == "Easy"):
            results.customgame.data_easy()
```

```
        elif(mode == "Medium"):
            results.customgame.data_medium()
        elif(mode == "Hard"):
            results.customgame.data_hard()
        print("=============================================================")
    else:
        import exitoption.exitop
        exitoption.exitop.exit()
        print("=============================================================")
```

data_qgame

```
def data_qgame():
    # A Welcome message to the user
    print('''
            ===============================================
           |                                               |
           |                Welcome to past game details!!!|
           |                                               |
            ===============================================
    ''')
    print("\t\t\t\t Game Play")
```

```
import mysql.connector
db = mysql.connector.connect(user="Useradmin",password="user123456",
                             host="127.0.0.1",database="game")
cursor = db.cursor()
# Extracting information from the table in the database using queries
cursor.execute("SELECT Name FROM quick_game")
data1 = cursor.fetchall()
cursor.execute("SELECT Total_Questions FROM quick_game")
data2 = cursor.fetchall()
cursor.execute("SELECT Correct_Answers FROM quick_game")
data3 = cursor.fetchall()
cursor.execute("SELECT Score FROM quick_game")
data4 = cursor.fetchall()
# Calling prettytable module to display a table to the user
from prettytable import PrettyTable
x = PrettyTable()
# Creating columns in the table
column_names = ["Name", "Total Questions","Correct Answers","Score"]
# Adding data from the database to the table
x.add_column(column_names[0],data1)
x.add_column(column_names[1],data2)
```

```
    x.add_column(column_names[2],data3)
    x.add_column(column_names[3],data4)
    # Displaying the table
    print(x)
    db.close()
```

data_easy

```
# Easy mode query to gather data from the database
def data_easy():
    import mysql.connector
    db = mysql.connector.connect(user="Useradmin",password="user123456",
                                 host="127.0.0.1",database="customgame")
    cursor = db.cursor()
    # Extracting information from the table in the database using queries
    cursor.execute("SELECT Name FROM easy_mode")
    data1 = cursor.fetchall()
    cursor.execute("SELECT Total_Questions FROM easy_mode")
    data2 = cursor.fetchall()
    cursor.execute("SELECT Correct_Answers FROM easy_mode")
    data3 = cursor.fetchall()
    cursor.execute("SELECT Score FROM easy_mode")
    data4 = cursor.fetchall()
```

```
    # Calling prettytable module to display a table to the user
    from prettytable import PrettyTable
    x = PrettyTable()
    # Creating columns in the table
    column_names = ["Name", "Total Questions","Correct Answers","Score"]
    # Adding data from the database to the table
    x.add_column(column_names[0],data1)
    x.add_column(column_names[1],data2)
    x.add_column(column_names[2],data3)
    x.add_column(column_names[3],data4)
    # Displaying the table
    print(x)
    db.close()
```

data_medium

```
# Medium mode query to gather data from the database
def data_medium():
    import mysql.connector
    db = mysql.connector.connect(user="Useradmin",password="user123456",
                                 host="127.0.0.1",database="customgame")
    cursor = db.cursor()
```

```
# Extracting information from the table in the database using queries
    cursor.execute("SELECT Name FROM medium_mode")
    data1 = cursor.fetchall()
    cursor.execute("SELECT Total_Questions FROM medium_mode")
    data2 = cursor.fetchall()
    cursor.execute("SELECT Correct_Answers FROM medium_mode")
    data3 = cursor.fetchall()
    cursor.execute("SELECT Score FROM medium_mode")
    data4 = cursor.fetchall()
    # Calling prettytable module to display a table to the user
    from prettytable import PrettyTable
    x = PrettyTable()
    # Creating columns in the table
    column_names = ["Name", "Total Questions","Correct Answers","Score"]
    # Adding data from the database to the table
    x.add_column(column_names[0],data1)
    x.add_column(column_names[1],data2)
    x.add_column(column_names[2],data3)
    x.add_column(column_names[3],data4)
    print(x)
    db.close()
```

data_hard

```
# Hard mode query to gather data from the database

def data_hard():

    import mysql.connector

    db = mysql.connector.connect(user="Useradmin",password="user123456",

                                 host="127.0.0.1",database="customgame")

    cursor = db.cursor()

    # Extracting information from the table in the database using queries

    cursor.execute("SELECT Name FROM hard_mode")

    data1 = cursor.fetchall()

    cursor.execute("SELECT Total_Questions FROM hard_mode")

    data2 = cursor.fetchall()

    cursor.execute("SELECT Correct_Answers FROM hard_mode")

    data3 = cursor.fetchall()

    cursor.execute("SELECT Score FROM hard_mode")

    data4 = cursor.fetchall()

    # Calling prettytable module to display a table to the user

    from prettytable import PrettyTable

    x = PrettyTable()

    # Creating columns in the table

    column_names = ["Name", "Total Questions","Correct Answers","Score"]
```

```
    # Adding data from the database to the table
    x.add_column(column_names[0],data1)
    x.add_column(column_names[1],data2)
    x.add_column(column_names[2],data3)
    x.add_column(column_names[3],data4)
    # Displaying the table
    print(x)
    db.close()
```

4.8 **No option

```
# User leaves the option without inputing anything or by inputing an invalid character
   else:
       import exitoption.exitop
       exitoption.exitop.exit()
print("================================================================")
```

exitop module(This also gets imported if user leaves past game details options unmarked)

```
# This is what's going to happen if the user chooses Exit option

def exit():

print('''
=====================================================================

      |                                                               |

      |                    follow the instructions carefully!!!       |

      |                                                               |
=====================================================================

      ''')
```

Conclusion

This program was developed to improve the calculation power of children who are studying and to make them actively involved in Mathematics. There are two game plays to this game”MathMania”, quick game and custom game. If the uses quick game in the beginning 5 questions will get generated based only on addition. And users’ correct answer count will get stored in a database with the score he/she has obtained. If the user chooses custom game he will be given a choice to choose among three difficulty levels called as easy, medium and hard. In custom game user will get as much as questions as he/she wants to answer. In the Easy mode questions will be generated based on only addition, medium mode based on addition and subtraction, hard mode based on addition, subtraction, multiplication and division. And in here also some data user has inserted will get stored inside of a database. And after each and every game play, game results will be displayed to the user.

If the user chooses past game details option, in the choices menu he/she will have to specify which game mode he/she is looking for. And according to his choice past game plays in each game mode will get displayed using the stored data in the database. This process will happen in the system until the user confirms to Exit.

References

Bodnar, J., 2020. *Python Prettytable Tutorial - Generating Tables In Python With Prettytable*. [online] Zetcode.com. Available at: <http://zetcode.com/python/prettytable/> [Accessed 8 April 2020].

PYnative. 2020. *Python Mysql Select From Table [Complete Guide]*. [online] Available at: <https://pynative.com/python-mysql-select-query-to-fetch-data> [Accessed 5 April 2020].

W3schools.com. 2019. *Python Mysql Create Table*. [online] Available at: <https://www.w3schools.com/python/python_mysql_create_table.asp> [Accessed 3 April 2020].

www.youtube.com. 2020. *Python - Prettytable Module*. [online] Available at: <https://www.youtube.com/watch?v=Ds00vFMzeOg> [Accessed 8 April 2020].

YouTube. 2020. *#36 Python Tutorial For Beginners | Global Keyword In Python | Global Vs Local Variable*. [online] Available at: <https://www.youtube.com/watch?v=QYUbLevwgDQ> [Accessed 3 April 2020].

YouTube. 2020. *Python And Mysql - Selecting And Getting Data*. [online] Available at: <https://www.youtube.com/watch?v=jA1GO6g_Rw0> [Accessed 5 April 2020].

YouTube. 2020. *Python Tutorial For Beginners (Complete Course)*. [online] Available at: <https://www.youtube.com/watch?v=IMGzzK9Wn4w> [Accessed 4 April 2020].

Youtube.com. 2020. *Converting Python Into Console Program - Youtube*. [online] Available at: <https://www.youtube.com/results?search_query=converting+python+into+console+program> [Accessed 5 April 2020].