Informatics Institute of Technology

Foundation Program for Higher Education

Module: DOC 334 Introduction to Programming – P2

Module Leader: Dr. Damitha Karunaratne

Date of Submission: 10/04/2020

Topic: Programming Individual assignment No.1

Group A

Student Name: Desira Wijesundara (2019024)

Acknowledgments

The following project would not have been possible if it wasn't for the intuitive and challenging tutoring of Dr. Damitha Karunaratne, and the excellent tutoring and mentorship of Mr. Nishan Saliya Harankahawa. My gratitude further extends to Ms. Kalpana Weerasinghe and Ms. Tharushi Amarasinghe for tutoring us the technical elements of working with the 'SQL' language, for it helped me immensely in this project as well.

Table of contents

Contents

List of tables and illustrations

1. Introduction

The process of constructing the most viable replica of the demonstration provided in the coursework specification for the mathematics game, was at times tedious yet substantial, for it forced me to go through stages of trial and error to reach the end goal. Heavily relying on all the past lecture material as well as tutorials, I was able to construct a functional mathematics game in due time. This game features a game menu which gives players the option to play a quick game, play a custom game, view past game details, or to exit the game, which will be discussed further down in the report. This project revolved around the ability to work with creating modules, importing custom modules as well as built-in modules, functions and using loops when necessary. 'XAMPP' and 'MySQL' were used in the process to store gameplay data, and ultimately the 'Pretty Table' module was used to display the results in an appealing manner to the user.

2. Problem Specification

From a simplified perspective this mathematics game should allow users to answer mathematical questions, which uses randomly generated numbers as well as randomly generated operators. It is clearly evident that the 'Random' module should be imported in order to start off the algorithm. This game should also allow the user to choose between the following options;

1) A quick game option which takes users' name, and asks the user 5 'addition' questions constructed out of randomly generated numbers in the range 0 to 10.
2) A custom game option which take users' name, difficulty level and number of questions he/she would like to answer and generate questions accordingly.
3) View past game play details.
4) Exit the game.

The following topics discuss the methods taken in order to convert the problem into an algorithm.

3. Solution

3.1. Main program

>>

```
q = 0
while q != 4:
   print()
   print('        Game Menu')
   print()
   print('1. Quick Game')
   print('2. Custom Game')
   print('3. Display past game details')
   print('4. Exit')
   print()

   option = input("Enter your option: ")

   if option == '1':
      import Quickgame.QG
      Quickgame.QG.quick()

   if option == '2':
      import Customgame.custom
      Customgame.custom.cm()

   if option == '3':
      print("Select from the following")
      print("1. Quick game details")
```

```
    print("2. Custom game details")
    user = input("Which details would you like to display? : ")
    if user == "1":
      import Pastdetails.QG_db
    if user == "2":
      import Pastdetails.CG_db

  if option == '4':
    q = 4
```

>>

The above shared algorithm functions as the main program, which imports modules according to users' preferred option. The 'q' variable was created globally to aid the algorithm to exit the program upon users' command. The first segment of the algorithm prints the game menu, the options available for the user; quick game, custom game, view past game details and exit as follows;

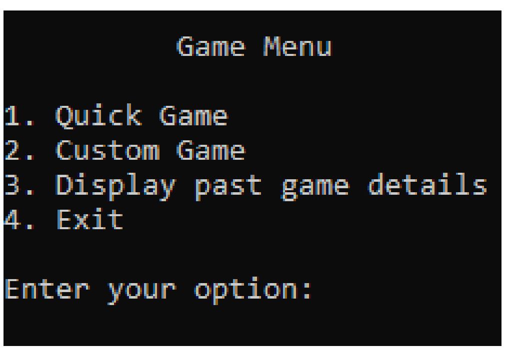

Figure 1/Menu

A variable called ‘option’ is created to take user input, on what option they prefer from the game menu. If the user enters number ‘1’, the ‘QG’ module from the ‘QuickGame’ is imported and the ‘quick’ function inside the module is called to the main program, giving the following output;

```
          Game Menu

1. Quick Game
2. Custom Game
3. Display past game details
4. Exit

Enter your option: 1
Enter your name:
```

Figure 2/Quick game

Once the user enters his/her name, the user will be provided with 5 randomly generated ‘addition’ questions, which he/she has to answer. Provided he/she answers all 5 of these questions, the user will be able to see their final score.

```
          Game Menu

1. Quick Game
2. Custom Game
3. Display past game details
4. Exit

Enter your option: 1
Enter your name: Desira
11 + 0 = 11
5 + 2 = 7
7 + 8 = 15
0 + 7 = 7
10 + 7 = 17

      Game results
Your name is  Desira
You played quickgame
You answered 5 questions
11 + 0 = 11(answer is 11)[Correct]
5 + 2 = 7(answer is 7)[Correct]
7 + 8 = 15(answer is 15)[Correct]
0 + 7 = 7(answer is 7)[Correct]
10 + 7 = 17(answer is 17)[Correct]
Your score is 100 %
```

Figure 3/Quick game – gameplay

If the user enters number ‘2’, the ‘custom’ module from the ‘CustomGame’ is imported and the ‘cm’ function inside the module is called to the main program, giving the following output;

```
          Game Menu

1. Quick Game
2. Custom Game
3. Display past game details
4. Exit

Enter your option: 2
Enter your name:
```

Figure 4/Custom game

When the user provides their name, he/she will be provided with the option to select the ideal difficulty for them, which being;

1) Easy mode which asks how many questions he/she needs and provides randomly generated ‘addition’ questions, using numbers in range 1 to 10 and finally printing the results as follows;

```
          Game Menu

1. Quick Game
2. Custom Game
3. Display past game details
4. Exit

Enter your option: 2
Enter your name: Desira
Following input is case sensitive
Enter preffered difficulty level(Easy/Medium/Hard): Easy
How many questions do you need? : 3
2 + 2 = 4
0 + 4 = 4
2 + 3 = 5

      Game results
Your name is Desira
You played the game on Easy mode
You answered 3  questions
2 + 2 = 4 (answer is 4) [Correct]
0 + 4 = 4 (answer is 4) [Correct]
2 + 3 = 5 (answer is 5) [Correct]
Your score is 100 %
```

Figure 5/Easy mode gameplay

2) Medium mode which functions just as the easy mode, with the only difference being that the randomly generated questions are not only limited to ‘addition’ questions but ‘subtraction’ questions as well, and the range of numbers are from 0 to 50. The output after completing the provided questions, is as follows;

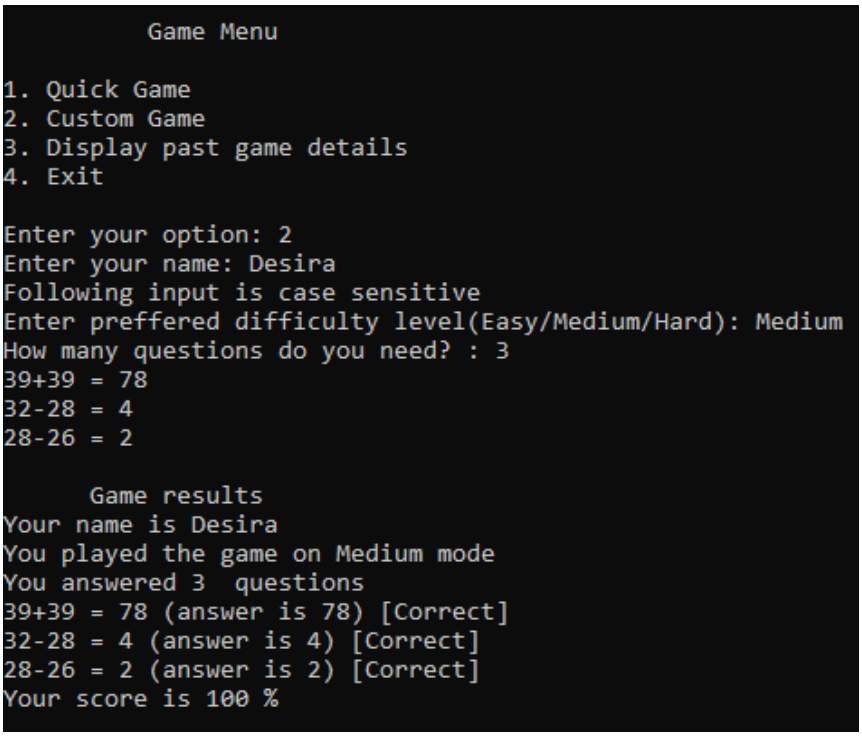

Figure 6/Medium mode gameplay

3) Hard mode which functions as above modes, with the exception being it containing ‘addition’, ‘subtraction’ and ‘multiplication’ questions and the randomly generated number range is from 0 to 100. A completed gameplay will look like the following;

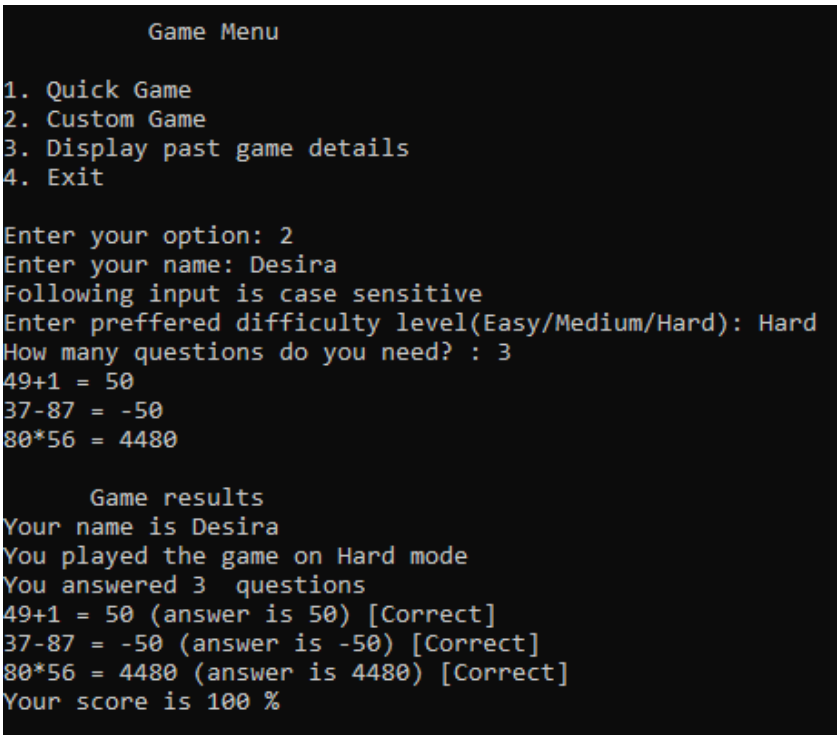

Figure 7/Hard mode gameplay

When the user enters the option ‘3’ it prompts the user with another menu, asking which game mode past game details he/she would like to view as follows;

```
          Game Menu

1. Quick Game
2. Custom Game
3. Display past game details
4. Exit

Enter your option: 3
Select from the following
1. Quick game details
2. Custom game details
Which details would you like to display? :
```

Figure 8/Past game details

Entering ‘1’ or ‘2’ will result in displaying of the previous gameplay results such as; the user name, correct questions, number of questions generated and the overall percentage as follows;

```
          Game Menu

1. Quick Game
2. Custom Game
3. Display past game details
4. Exit

Enter your option: 3
Select from the following
1. Quick game details
2. Custom game details
Which details would you like to display? : 1
+--------+---------+-----------------+------------+
|  Name  | Correct | Total_questions | Percentage |
+--------+---------+-----------------+------------+
|  Ddd   |    5    |        5        |    100     |
| Desira |    5    |        5        |    100     |
+--------+---------+-----------------+------------+
```

Figure 9/Quick game details

```
          Game Menu

1. Quick Game
2. Custom Game
3. Display past game details
4. Exit

Enter your option: 3
Select from the following
1. Quick game details
2. Custom game details
Which details would you like to display? : 2
+--------+---------+-----------------+------------+
|  Name  | Correct | Total_questions | Percentage |
+--------+---------+-----------------+------------+
| Desira |    3    |        3        |    100     |
|  Des   |    2    |        3        |     66     |
|  ddd   |    2    |        3        |     66     |
| Desira |    3    |        3        |    100     |
| Desira |    3    |        3        |    100     |
| Desira |    3    |        3        |    100     |
+--------+---------+-----------------+------------+
```

Figure 10/Custom game details

Once the user enters ‘4’, the global variable ‘q’ gets assigned with ‘4’ and the while loop breaks, terminating the program, as shown below;

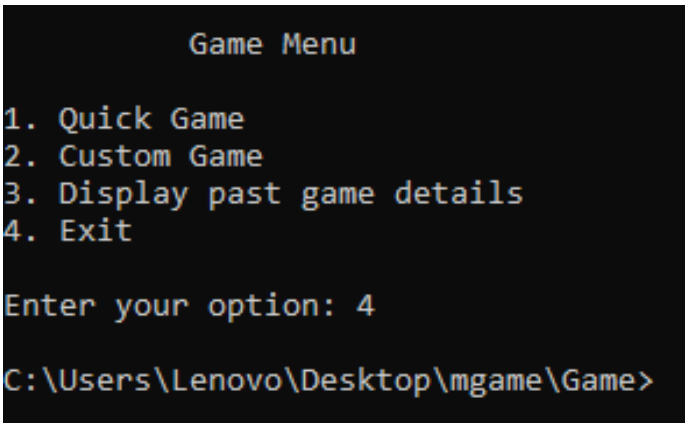

Figure 11/exit

3.2. Quick game

```
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
def quick():

  import random
  import mysql.connector

  score = 0
  ques_lst = []

  name = input("Enter your name: ")

  for i in range(5):
    num1 = random.randint(0,11)
    num2 = random.randint(0,11)

    ans = num1 + num2

    ques = int(input(str(num1)+' + '+str(num2)+' = '))

    if ques == ans:
      score += 1
      ques_lst.append(str(num1)+' + '+str(num2)+' = '+str(ques)+'(answer is '+str(ans)+')[Correct]')
    else:
      ques_lst.append(str(num1)+' + '+str(num2)+' = '+str(ques)+'(answer is '+str(ans)+')[Incorrect]')
  print()
  print('      Game results')
  print('Your name is ',name)
```

```
print('You played quickgame')
print('You answered 5 questions')
for val in range(5):
    print(ques_lst[val])

print('Your score is',int((score/5)*100),'%')

Dict = {"host":"localhost",
        "database":"game_db",
        "user":"root",
        "password":""}

db = mysql.connector.connect(**Dict)

cursor = db.cursor()

sql = "INSERT INTO quickgame (name,correct,Total_questions,percentage) VALUES (%s,%s,%s,%s)"
plyr = (name,score,"5",int((score/5)*100))
cursor.execute(sql,plyr)

db.commit()

db.close
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

The above algorithm starts by declaring the entire module as a function, namely 'quick', importing two modules called 'random' and 'mysql.connector', a variable called score which is assigned with the value 0 and an empty list called 'ques_lst' which will be used to append gameplay details such as users' answer, the correct answer, and whether the users' answer is correct or incorrect,

further down the lines of code. ‘name’ variable is created to store users name, and in the next segment a ‘for’ loop is used to print 5 questions. ‘num1’ and ‘num2’ variables are used to store the randomly generated numbers in between the range 0 to 11, and ‘ans’ variable stores the sum of the two randomly generated two values, which will be used as the correct answer. The program will use the two randomly generated values to ask an ‘addition’ question from the user, take the input, convert it into an integer value and store the value in the ‘ques’ variable. If the user input and the real answer is equal, the answer is considered correct. If the answer is correct the score will be increased in value by 1, and the ‘ques_lst’ will be appended with the users’ answer, the correct answer, and that the user answer is correct. If the user answer is incorrect the score will remain the same and everything will append as mentioned above with the exception of mentioning that the answer is incorrect in the end.

Finally the results will be printed out which includes, the users’ name, which game mode he/she played, how many questions were played, each element in the ‘ques_lst’, and finally the user score percentage which was calculated by dividing the correct answers by 5, multiplying it by 100 and converting it into and integer.

The rest of the code will be using the ‘mysql.connector’ module to store data into the databases created in the local host. A variable called ‘Dict’ will store a dictionary of details to connect to the local host. Then ‘db’ variable will store the connection to local host, and a cursor object is created to navigate through the database. The data will be inserted into the local host table; ‘quickgame’ inside the ‘game_db’ database, and the changes will be committed.

3.3. Custom game

```
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

def cm():
   import random
   import mysql.connector

   def results(name,mode,ques):
      print()
      print('     Game results')
      print('Your name is',name)
      print('You played the game on',mode,'mode')
      print('You answered', ques,' questions')

   Dict = {"host":"localhost",
        "database":"game_db",
        "user":"root",
        "password":""}

   score = 0
   name = ''
   ques_lst = []
   ops_med = ['+','-']
   ops_hard = ['+','-','*']

   name = input("Enter your name: ")

   print('Following input is case sensitive')
   dif = input("Enter preffered difficulty level(Easy/Medium/Hard): ")
```

```
    questions = int(input("How many questions do you need? : "))

    if dif == 'Easy':

        for i in range(questions):
            num1_e = random.randint(0,11)
            num2_e = random.randint(0,11)

            ans = num1_e + num2_e

            ques = int(input(str(num1_e) + ' + ' + str(num2_e) + ' = '))

            if ques == ans:
                score += 1
                ques_lst.append(str(num1_e)+' + '+str(num2_e)+' = '+str(ques)+' (answer is ' + str(ans) + ')
[Correct]')

            else:
                ques_lst.append(str(num1_e)+' + '+str(num2_e)+' = '+str(ques)+' (answer is ' + str(ans) + ')
[Incorrect]')

        results(name,dif,questions)

        for val in range(questions):
            print(ques_lst[val])

        print('Your score is',int((score/questions)*100),'%')
```

```
        db = mysql.connector.connect(**Dict)

        cursor = db.cursor()

        sql = "INSERT INTO customgame (name,correct,Total_questions,percentage) VALUES
(%s,%s,%s,%s)"
        plyr = (name,score,questions,int((score/questions)*100))
        cursor.execute(sql,plyr)

        db.commit()

        db.close



    if dif == 'Medium':
        for i in range(questions):

            num1_m = random.randint(0,51)
            num2_m = random.randint(0,51)

            operator = random.choice(ops_med)

            if operator == '+':
                ans = num1_m + num2_m

            if operator == '-':
                ans = num1_m - num2_m

            ques = int(input(str(num1_m) + str(operator) + str(num2_m) + ' = '))
```

```
        if ques == ans:
            score += 1
            ques_lst.append(str(num1_m)+ str(operator) +str(num2_m)+' = '+str(ques)+' (answer is ' + str(ans) + ') [Correct]')

        else:
            ques_lst.append(str(num1_m)+ str(operator) +str(num2_m)+' = '+str(ques)+' (answer is ' + str(ans) + ') [Incorrect]')

    results(name,dif,questions)

    for val in range(questions):
        print(ques_lst[val])

    print('Your score is',int((score/questions)*100),'%')

    db = mysql.connector.connect(**Dict)

    cursor = db.cursor()

    sql = "INSERT INTO customgame (name,correct,Total_questions,percentage) VALUES (%s,%s,%s,%s)"
    plyr = (name,score,questions,int((score/questions)*100))
    cursor.execute(sql,plyr)

    db.commit()

    db.close
```

```
    if dif == 'Hard':
        for i in range(questions):

            num1_h = random.randint(0,101)
            num2_h = random.randint(0,101)

            operator = random.choice(ops_hard)

            if operator == '+':
                ans = num1_h + num2_h

            if operator == '-':
                ans = num1_h - num2_h

            if operator == '*':
                ans = num1_h * num2_h

            ques = int(input(str(num1_h) + str(operator) + str(num2_h) + ' = '))

            if ques == ans:
                score += 1
                ques_lst.append(str(num1_h)+ str(operator) +str(num2_h)+' = '+str(ques)+' (answer is ' +
str(ans) + ') [Correct]')

            else:
                ques_lst.append(str(num1_h)+ str(operator) +str(num2_h)+' = '+str(ques)+' (answer is ' +
str(ans) + ') [Incorrect]')
```

```
    results(name,dif,questions)

    for val in range(questions):
        print(ques_lst[val])

    print('Your score is',int((score/questions)*100),'%')

    db = mysql.connector.connect(**Dict)

    cursor = db.cursor()

    sql = "INSERT INTO customgame (name,correct,Total_questions,percentage) VALUES
(%s,%s,%s,%s)"
    plyr = (name,score,questions,int((score/questions)*100))
    cursor.execute(sql,plyr)

    db.commit()

    db.close
```

>>

The above algorithm functions almost identical to the code mentioned in the quick game topic, with exceptions being that it asks users the amount of questions they like to answer, and on medium and hard settings, numbers are generated from 0 to 50 and 0 to 100 respectively. Lists were used to contain operators, such as addition, subtraction and multiplication, which were used in the process of generating random questions, with the aid of the ‘choice’ function inside the ‘random’ module. Finally, the data will be inserted into the local host table; ‘customgame’ inside the ‘game_db’ database, and the changes will be committed.

3.4. Display past game details

Both of the following sub topics shows the algorithms that were used to display the past game results in a tabular format. Pretty table module had to be installed into python firsthand, before it was used in the code. With the usage of the 'from_db_cursor' function inside the 'prettytable' module, it was possible to find a way to display the results that were stored in the database tables, in tabular format instead of the default list format.

3.4.1. Quick game details

>>>

```
import mysql.connector
from prettytable import from_db_cursor

#create a dictionary to store the data of localhost
Dict = {"host":"localhost",
    "database":"game_db",
    "user":"root",
    "password":""}
db = mysql.connector.connect(**Dict)

cursor = db.cursor()

cursor.execute("SELECT * FROM quickgame")

data = from_db_cursor(cursor)

print(data)
db.close()
```

>>>

3.4.2. Custom game details

>>

```
import mysql.connector
from prettytable import from_db_cursor


Dict = {"host":"localhost",
       "database":"game_db",
       "user":"root",
       "password":""}


db = mysql.connector.connect(**Dict)


cursor = db.cursor()


cursor.execute("SELECT * FROM customgame")


data = from_db_cursor(cursor)


print(data)


db.close()
```

>>

Conclusion

The final outcome of this project is a functional math game which allows a user to play a quick game which prompts 5 randomly generated ‘addition’ questions, a custom game with three difficulty levels, each varying from the range of numbers, and operators used to generate any number of questions the user wants, a data retrieval option which prints past gameplay details using the ‘SQL’ language and a fourth option which terminates the program.

References

Stackoverflow.com 2020. [online] Available at: < https://stackoverflow.com/> [Accessed March 2020]

W3schools.com, 2020. [online] Available at: <https://www.w3schools.com/> [Accessed March 2020]