



Informatics Institute of Technology

Foundation Program in Higher Education

Module: DOC334 Introduction to Programing P2

Module Leader: Dr Damitha Karunaratne

Assessment Type: Math Quiz

Date of Submission: 10-04-2020

Student Name: Nemsara Sasvitha Mapalana Gamage

Student ID: 2019238

Abstract

This report is aimed to help the user understand how to create a math quiz program as there are many ways that this program can be made, The methods used here is an easier and more detailed approach to make a math quiz using modules, function, packages and saving the data of the users results to database in MySQL and being able to run the program on the command console.

Contents

| | |
|--|----|
| Abstract..... | 2 |
| Acknowledgement..... | 4 |
| 1. Introduction..... | 5 |
| 1.1 Description of the Problem | 5 |
| 1.2 Description of Quick game | 5 |
| 1.3 Description of Custom game..... | 5 |
| 1.3.1 Easy level..... | 5 |
| 1.3.2 Medium level..... | 6 |
| 1.3.3 Hard level | 6 |
| 1.4 Description of past games results..... | 6 |
| 1.5 Description of exiting the program | 6 |
| 2. Solution..... | 7 |
| 2.1 Main program..... | 7 |
| 2.2 Sources | 8 |
| 2.3 Game menu | 8 |
| 2.4 Loop | 9 |
| 3. Test cases | 35 |
| 4. Conclusion..... | 36 |

Acknowledgement

I am highly grateful to IIT for their guidance and supervision as well as for providing me with the necessary information regarding the project. I would like to thank Dr Damitha Karunaratne who was my lecture on this module and Mr. Nishan Hrankahawa who was my tutor on this module for providing me the knowledge and teaching me the need skill to undertake this project and succeed in it. I would also like to thank my parents as well as those who have assisted me directly or indirectly with this project.

1. Introduction

1.1 Description of the Problem

This program is a math quiz game that is written in the python program and made to run in the command console. It has four important parts that are

- Quick game
- Custom game
- Displaying the past game results
- Exiting the program

1.2 Description of Quick game

This quick game is an easy mode where it only contains the addition math operator and only consist of five questions. The user need to input his name and after he has done that he will get the five questions, the range of those questions will be between 0 to 11 and after he has done answering the those question he will get his results which will display his name and the number of question he has gotten correct, the number of question he answered and final his score in percentage.

1.3 Description of Custom game

This is divided into 3 levels:

- Easy
- Medium
- Hard

1.3.1 Easy level

The easy level is the same as the quick game but we ask the user to input the number of questions that he would like to answer as we use the addition math operator here as well, so after the user has inputted the number of questions he want the program will give question based on the number of questions and the range is still going to be the same as it is in the quick game that is 0 to 11. When the results are displayed to the user it will show the user name, the difficulty he played on, the questions he got correct, the number of questions he has inputted and final the score in percentage.

1.3.2 Medium level

The medium level is the same as the easy level but we need to use both addition and subtraction math operators and the user get to input the number of questions he wants and the range of those question will be from 0 to 51. So, after the user answered all the question, he will get the same results as in easy level that is the user name, the difficulty he played on, the questions he got correct, the number of questions he has inputted and final the score in percentage.

1.3.3 Hard level

The hard level is the same as the medium level but for the hard level we need the addition, subtraction and the multiplication math operator and the range here will be from 0 to 101 because we need it make it hard for the user. So, after the user answered all the question, he will get the same results as in medium level that is the user name, the difficulty he played on, the questions he got correct, the number of questions he has inputted and final the score in percentage

1.4 Discription of past games results

To show the user there past game results we need to create a database and we use MySQL for that. Then there need to be a connection to the database in order to the store the information of the user results to the database.

1.5 Discription of exiting the program

The three need to be a way for the user to exit the program.

2. Solution

2.1 Main program

```
import sys
import mysql.connector
import quickgame.quickgame
import customgame.mode1.easy
import customgame.mode2.medium
import customgame.mode3.hard

# inserting the gamemenu to a function
def gamemenu():
    print()
    print("        Game menu")
    print("a) Quick Game")
    print("b) custom Game")
    print("c) Display Past Game Details")
    print("d) Exit")
    print()
    return str(input("Enter your option:"))#users option to choose one of them
    print()
    print()

while True:
    user_op=gamemenu()
    #if user inputs "a" for quick game
    if (user_op == "a"):
        print()
        print("        Quick Game")
        print()
        quickgame.quickgame.quick()

    #if user inputs "b" for custom game
    elif (user_op == "b"):
        print()
        print("        Custom Game")
        print()
        diff=str(input("Enter difficulty (easy/medium/hard)= "))
        if (diff == "easy"):
            print()
            customgame.mode1.easy.ez()
        elif (diff == "medium"):
            print()
            customgame.mode2.medium.med()
        else:
            (diff == "hard")
```

```

        print()
        customgame.mode3.hard.hard()

#if user inputs "c" to view past game results
elif (user_op == "c"):
    print()
    print ("    Past Game Detail")
    print()
    ask=str(input("View past score for Quick Game or Custom Game: "))
    if(ask == "Quick game"):
        import database.qgamedatabase
    else:
        (ask == "Custom game")
        import database.cgamedatabase

#if user inputs "d" to exit
elif(user_op == "d"):
    exit()

```

2.2 Sources

In the main program we start with importing all the sources:

```

import sys
import mysql.connector
import quickgame.quickgame
import customgame.mode1.easy
import customgame.mode2.medium
import customgame.mode3.hard

```

we have to first import sys because this is a console command program and it cannot function the command console without import sys. The rest of the imports are the packages and modules for the program.

2.3 Game menu

After importing all the sources, we go to the game menu that is within a function:

```

def gamemenu():
    print()
    print("        Game menu")
    print("a) Quick Game")
    print("b) custom Game")
    print("c) Display Past Game Details")
    print("d) Exit")
    print()

```



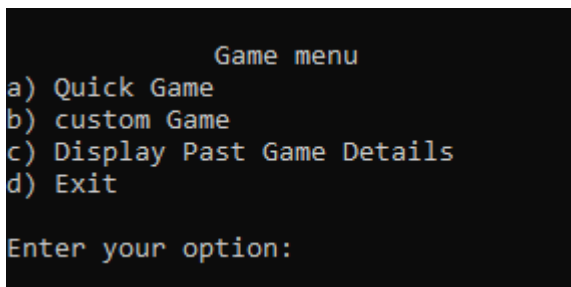
```

return str(input("Enter your option:"))
print()
print()

```

I named the def function as “gamemenu” and within the function I first printed the heading “Game menu”, then I gave the four options to the user, option “a” is for quick game, option “b” is for custom game, option “c” is to display the past game results and option “d” is to let the user exit the program. There is “print()” there to give some space when the program ask the user for there input. There is an option for the user to input his selection from all four of those options.

2.3.1 Output



```

                Game menu
a) Quick Game
b) custom Game
c) Display Past Game Details
d) Exit

Enter your option:

```

Figure 1 Output of the Game menu

2.4 Loop

So after the user inserts one out of the four option the program will start repeating on a loop, I used the “while True” statement here because it will run infinite number of times even after the user is done with the option that the user has chosen, it will go back to the game menu and will ask for the user input again and if the user input is an invalid option it will bring him back to the game menu. I also assigned the “gamemenu” to “user_op” so that the loop can identify what option out of the four the user has inputted.

```

while True:
    user_op=gamemenu()

```

Inside the “While True” statement we use the if condition to see if “user_op” is equal to one of the four options given

2.4.1 If users’ input is “a”:

```

if (user_op == "a"):
    print()
    print("        Quick Game")
    print()
    quickgame.quickgame.quick()

```

If the user's input is "a" the if condition will be executed and it will start by first printing the title which is "Quick game" and then it will go to execute quickgame.quickgame.quick(), now for the quick game I have created it in another python program and made it into a module and put it inside of a package.






| | | | |
|---|-------------|-------------------|-------------|
|  | __pycache__ | 4/3/2020 7:51 PM | File folder |
|  | customgame | 4/2/2020 12:08 AM | File folder |
|  | database | 4/4/2020 5:40 PM | File folder |
|  | quickgame | 4/3/2020 5:55 PM | File folder |
|  | main | 4/5/2020 6:08 PM | PY File |

Figure 2 All the program



| | | | |
|---|-------------|-------------------|-------------|
|  | __pycache__ | 4/4/2020 5:37 PM | File folder |
|  | quickgame | 4/4/2020 10:44 AM | PY File |

Figure 3 Inside the quickgame package

Quick game python program:

```
import mysql.connector
```

```
def quick():
```

```
    import random
```

```
#variables
```

```
    score=0
```

```
    rand=0
```

```
    update=[]
```

```
#asking the users name
```

```
    name=str(input("Enter name:"))
```

```
#main program
```

```
    for i in range(5):
```

```
        n1=random.randint(0,11)#importing 2 random values
```

```
        n2=random.randint(0,11)
```

```
#putting the whole question in a single line
```

```
    question=int(input(str(n1)+"+"+str(n2)+"="))
```

```
#getting the real answer to see if it matches with the question
```

```
    ans=n1+n2
```

```
#checking if the answers match the users input
```

```
    if question == ans:
```

```
        #keeping track of the score
```

```
        score = score +1
```

```
        update.append(str(n1)+' + '+str(n2)+' = '+str(question)+'(Answer is '+str(ans)+'')
```

```
[correct]')
```

```
    else:
```

```
        update.append(str(n1)+' + '+str(n2)+' = '+str(question)+'(Answer is '+str(ans)+'')
```

```
[incorrect]')
```

```
for val in range(5):  
    print(update[val])
```

```
#printing the results  
print("  Results")  
print("Your name is: ",name)  
print("you have got",score,"/5 correct")  
print("your scored : ",(score/5)*100,"/100")
```

Solution:

The quick game program is written inside a function called “quick()”. Inside the quick() I first imported random as we need random numbers to generate and give it to the users, then I created the variables which are score=0, rand=0, update=[] and name. The score variable is there to show the user his score when he answers the given question, then rand=0 is there for random number as we brought it in when we import random, then we have update=[] this is assigned to a list because of printing the correct answer if the user input is equal to the answer of the question and also a variable for the user to input his name.

The main program for quick game is done in a for loop and the range is five because it is a quick game, first I assign two variables that will give random number every time the program is executed and for that we use “range.randint”, the variables are n1 and n2 and the range of the random number will be out of 0 to 11 because we need to make the question very easy and then I create a variable called “question” which will take the two variable that are n1 and n2 and the operator that is used for the questions in the quick game which will be addition and combine it into a single line so that the user can input his answer to the randomly generated question ,I used “int” because we want the user input to be numbers not a string.

Then I had to create a method to check whether the answer is right or wrong and I did it by first creating a variable called “ans” which means answer and within that variable I inserted n1 and n2 so when random numbers are generated it will also get the answer of the random numbers generated so later it can check if the user input is correct or wrong, then we use an if condition to check whether the answer is right or wrong. So, in the if condition if the “question” which is the user’s input is equal to the “ans” which is the answer of the randomly generate question, if it is equal then the program will tell you the answer is correct and it will also show the question with the users input this is possible because I used append to insert it into the “update=[]” list and the “score=score+1” is there to count the number of question the user got correct, if the answer is not equal the program will tell you that the answer is incorrect and show you the real answer along with the users answer when it is uploaded to the list. Now to display the list I use a for loop with same range five because that is the number of question that the user will get and the it will print the list.

When showing the results it will only show the user name, the users score out of how many question he has go correct and then the score we have to show in percentage and for that I have

used $(\text{score}/5)*100$ so it will give us the percentage of the users score I put divided by five because we need to divide it out of the number of questions the user answered.

Output:

```

                Game menu
a) Quick Game
b) custom Game
c) Display Past Game Details
d) Exit

Enter your option:a

                Quick Game

Enter name:Nemsara
6+7=8
0+4=4
4+4=8
4+5=9
3+1=4
6 + 7 = 8(Answer is 13)[Incorrect]
0 + 4 = 4(Answer is 4)[Correct]
4 + 4 = 8(Answer is 8)[Correct]
4 + 5 = 9(Answer is 9)[Correct]
3 + 1 = 4(Answer is 4)[Correct]
    Results
Your name is: Nemsara
you have got 4 /5 correct
your scored : 80.0 /100
```

Figure 4 Output of the quick game

2.4.2 If users' input is "b"

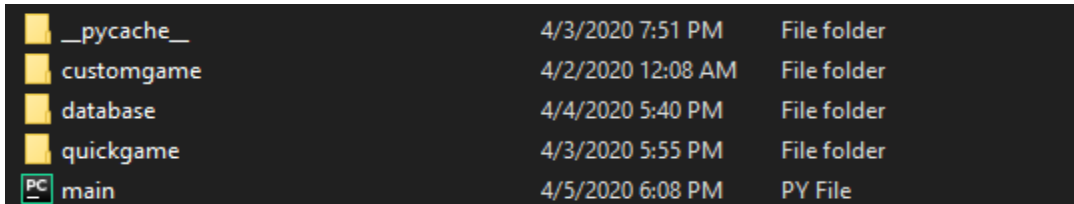
```
elif (user_op == "b"):
    print()
    print("        Custom Game")
    print()
    diff=str(input("Enter difficulity (easy/medium/hard)= "))
```

If user input is "b" the program will execute the elif statement and it will begin with printing the title of the game that is custom game and then I created a variable called "diff" which is short for difficulty , I did this because custom game has three levels easy, medium and hard so in order to find out what difficulty the user wants to play the custom game this command will run. So, after he has made his choice the program will be go on into an if condition depending on the user's input.

A. If user input is “easy”

```
if (diff == "easy"):
    print()
    customgame.mode1.easy.ez()
```

If the user’s input is “easy” it will run customgame.mode1.easy.ez(),



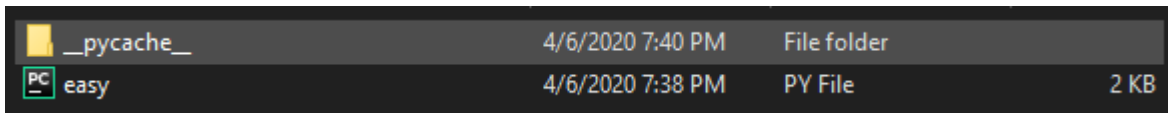
| | | |
|-------------|-------------------|-------------|
| __pycache__ | 4/3/2020 7:51 PM | File folder |
| customgame | 4/2/2020 12:08 AM | File folder |
| database | 4/4/2020 5:40 PM | File folder |
| quickgame | 4/3/2020 5:55 PM | File folder |
| PC main | 4/5/2020 6:08 PM | PY File |

Figure 5 All the folders



| | | |
|-------|-------------------|-------------|
| mode1 | 3/31/2020 9:05 PM | File folder |
| mode2 | 3/31/2020 9:05 PM | File folder |
| mode3 | 3/31/2020 9:05 PM | File folder |

Figure 6 Inside the customgame package



| | | | |
|-------------|------------------|-------------|------|
| __pycache__ | 4/6/2020 7:40 PM | File folder | |
| PC easy | 4/6/2020 7:38 PM | PY File | 2 KB |

Figure 7 Inside mode1 where the easy level model is contained

Model1 contains the module for the easy difficulty program which will execute when the user inserts the input.

Easy difficulty python program :

```
def ez():
    import random
    #variables
    score=0
    rand=0
    update=[]
    name=str(input("Enter name: "))
    times=int(input("How many questions would you like? = "))
    print()
```

```

#main program
for i in range(times):
    n1=random.randint(0,11)#importing 2 random values
    n2=random.randint(0,11)
    #putting the whole question in a single line
    question=int(input(str(n1)+"+"+str(n2)+"="))
    #getting the real answer to see if it matches with the question
    ans=n1+n2
    #checking if the answers match user input
    if question == ans:
        #keeping track of the score
        score = score +1
        update.append(str(n1)+' + '+str(n2)+' = '+str(question)+'(Answer is '+str(ans)+'
[correct]')
    else:
        update.append(str(n1)+' + '+str(n2)+' = '+str(question)+'(Answer is '+str(ans)+'
[incorrect]')

for val in range(times):
    print(update[val])

#printing the results
print()
print("  Results")
print("Your name is: ",name)
print("You played on: easy mode")
print("You choosen this many questions: ",times)
print("you have got",score,"/",times,"correct")
print("your scored : ",(score/times)*100,"/100")

```

Solution:

I first put the whole program into a function called ez().I imported random first then I created five variables “score=0” for the score, ”rand=0” for random numbers, ”update=[]” for making a list, “name” to input the users name and “times” for the users to input the no of questions that he or she might want.

For the main program I used a for loop with range being “times” because the program needs to display the question depending on the number of times the user wants. Then I created two variables that will give me two random number for that we use “range.randint” and the variables are n1 and n2 and the range of the random number will be from 0 to 11 because we need to make the question easy for the user, after this I made a variable called “question” to input n1 and n2 and make it into a question in a single line so that the user can input his answer to the randomly generated questions and the operator used for the easy difficulty is addition so it is hardcoded into the line.

So after the user is done inputting his answers there need to be a way to check whether it is correct or incorrect so a variable called “ans” was created and in there I insert n1 and n2 so that the randomly generated numbers that were given to the user has the answer and can check if the answer that the user inputted is correct or incorrect. Now to display if the answer is correct or incorrect we use an if condition so if the question is equal to the user input it will append the question into the update list and add a few more line showing the answer and telling the user it is correct and it will also show the question along with the user answer so that the user can verify his answer, all within the question variable that is appended to a list or if the question is not equal to ans it will do the same thing and append it to the list and will display the question along with the users answer and give the correct answer and will print a message that the answer is incorrect all within the question variable that is appended to a list.

So, to print the list where the results are kept, I created a for loop and the range will be “times” because it will print the results depending on the number of questions and then I print the update variable where all the results are kept in a list.

Then I print the results of the user, first we print the name, then we print the difficulty that he played on, then I show the number of questions that he asked for, then I print the score and out of how many question that he asked for was correct and finally I give his score in percentage but for custom game I did “(score/times)*100” because we need to get the percentage out of the many questions that are there so “score” will divide “times” which are the number of question and then give the percentage of the user score out of hundred.

Output:

```
Game menu
a) Quick Game
b) custom Game
c) Display Past Game Details
d) Exit

Enter your option:b

Custom Game

Enter difficulty (easy/medium/hard)= easy

Enter name: Nemsara
How many questions would you like? = 5

0+3=3
7+8=9
2+8=10
0+0=0
6+4=10
0 + 3 = 3(Answer is 3)[Correct]
7 + 8 = 9(Answer is 15)[Incorrect]
2 + 8 = 10(Answer is 10)[Correct]
0 + 0 = 0(Answer is 0)[Correct]
6 + 4 = 10(Answer is 10)[Correct]

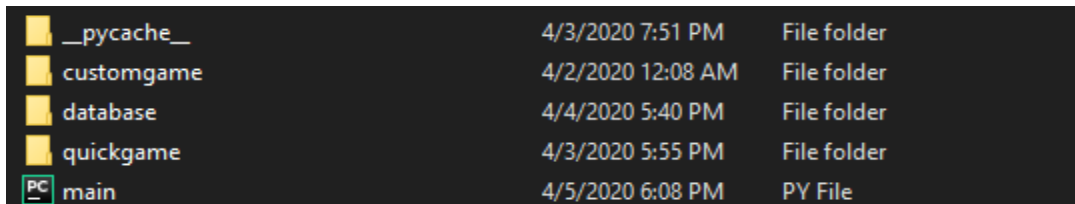
Results
Your name is: Nemsara
You played on: easy mode
You chosen this many questions: 5
you have got 4 / 5 correct
your scored : 80.0 /100
```

Figure 8 Easy level output in custom game

B. If user inputs “medium”

```
elif (diff == "medium"):
    print()
    customgame.mode2.medium.med()
```

If users’ option is “medium” the elif statement would be executed and it will run `customgame.mode2.medium.med()`.









| | | | |
|---|-------------|-------------------|-------------|
|  | __pycache__ | 4/3/2020 7:51 PM | File folder |
|  | customgame | 4/2/2020 12:08 AM | File folder |
|  | database | 4/4/2020 5:40 PM | File folder |
|  | quickgame | 4/3/2020 5:55 PM | File folder |
|  | main | 4/5/2020 6:08 PM | PY File |

Figure 9 All the folders





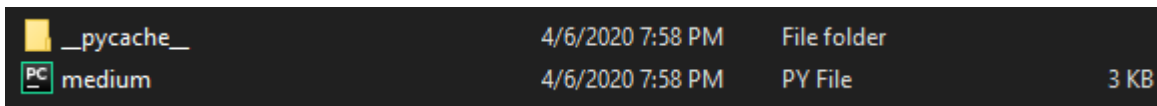
| | | | |
|---|-------|-------------------|-------------|
|  | mode1 | 3/31/2020 9:05 PM | File folder |
|  | mode2 | 3/31/2020 9:05 PM | File folder |
|  | mode3 | 3/31/2020 9:05 PM | File folder |

Figure 10 Inside the customgame package





| | | | |
|---|-------------|------------------|--------------|
|  | __pycache__ | 4/6/2020 7:58 PM | File folder |
|  | medium | 4/6/2020 7:58 PM | PY File 3 KB |

Figure 11 Inside mode2 where the medium level model is contained

Mode 2 contains the module for the medium difficulty which will execute when the user inserts his input

Medium difficulty python program:

```
def med():
    import random
    #variables
    score=0
    rand=0
    update=[]
    #creating list with the operators in them
    op=["+", "-"]
    name=str(input("Enter name: "))
    times=int(input("How many questions would you like? = "))
    print()
    #main program
```



```

for i in range(times):
    n1=random.randint(0,51)#importing 2 random values
    n2=random.randint(0,51)
    operators=random.choice(op)# used to chooose a random value from the "op"
    #putting the whole question in a single line
    question=int(input(str(n1)+str(operators)+str(n2)+"="))
    #getting the real answer to see if random operator is "+"
    if(operators == "+"):
        ans=n1+n2
    #getting the real answer to see if random operator is "-"
    if(operators == "-"):
        ans=n1-n2
    #checking if the answers match user input
    if question == ans:
        #keeping track of the score
        score = score +1
        update.append(str(n1)+str(question)+str(n2)+' = '+str(question)+'(Answer is '+str(ans)+' ) [correct]')
    else:
        update.append(str(n1)+str(question)+str(n2)+' = '+str(question)+'(Answer is '+str(ans)+' ) [incorrect]')

for val in range(times):
    print(update[val])

#printin the results
print()
print("  Results")
print("Your name is: ",name)
print("You played on: medium mode")
print("You choosen this many questions: ",times)
print("you have got",score,"/",times,"correct")
print("your scored : ",(score/times)*100,"/100")

```

Solution:

The medium difficulty is inside a function called `med()`, inside the function we start by importing `random` and then we create the variables like we did for the easy difficulty but here we add an extra variable called `"op"` which is short for operator for custom game we need both addition and subtraction for the questions so I created a variable that will hold the two operators in a list.

For the main program we use the for loop with `"times"` as range because it will run in the loop depending on the number of questions that the user inputs, after that again we create two variables that will give us random numbers for that we use `"range.randint"` and the two variables are `n1` and `n2` but the range of the random number will be from 0 to 51 because we need to make it a bit difficult for the user, and I created a new variable there called `"operators"` to store the math operators that we have created in `"op"` variable, I did this because we need to get questions both in addition and subtraction and the `"operators"` variable will give different types of operators this is possible because inside the `"operators"` variable we used `"random.choice(op)"` now what this does is it will choose the different math operator that is saved in the `"op"` variable it will choose addition or subtraction every time the program reruns depending on the amount of question the user wants. Then I created the `"question"` variable to insert the question in a single line but instead of the using the default add operator I put the `"operators"` variable there so it will give a different math operator each time the program reruns.

Now for getting the answer of the random number that were given to the user and the random operator that is used as well, I created an if condition to check whether the answer is correct if the question is addition, so if the operators which is the variable that is assigned to `"op"` that has different math operators inside the string is equal to addition then the answer is `n1+n2` so for the answer I created a variable called `"ans"` that will give the answer if the question is addition. But if the question is subtraction, we have to create another if condition to check if operators is equal to subtraction and then the answer will be `n1-n2` which will be assigned to the variable `"ans"`., both of these has one thing in common they are both there to give the answer for the random operator that is generated in the question so we can display it to the user.

For finding out whether the answer is correct or incorrect we use an if condition, so if the question is equal to the answer the score gets added by one and then it will be appended to the update list where the answer is stored but here we not only append the `"n1"` and `"n2"` variable but also the `"operators"` variable so it will show the user the number of the two variable and the operator that was answered, along with the real answer to show that the user answer is correct and a message telling him that his answer is correct, or if the question not equal to the answer the else statement will be executed and the same thing will repeat where it will be appended to the update list showing the user the two random number and the operator used and showing his answer and then showing the real answer and telling the user that his answer was incorrect. Then we use a for loop to print the list to show the user whether his inputs were right or wrong

For the results I started by print the heading which is `"results"` , then the name, then the level of difficulty the user had played on which is medium, then the number of questions the user has chosen, then showing him how much he has scored out of the total question and finally showing him his score in percentage out of a hundred

Output:

```
Game menu
a) Quick Game
b) custom Game
c) Display Past Game Details
d) Exit

Enter your option:b

Custom Game

Enter difficulty (easy/medium/hard)= medium

Enter name: Jack
How many questions would you like? = 5

13-41=-1
12-7=5
41-25=16
21-13=6
29-9=20
13-41 = -1(Answer is -28)[Incorrect]
12-7 = 5(Answer is 5)[Correct]
41-25 = 16(Answer is 16)[Correct]
21-13 = 6(Answer is 8)[Incorrect]
29-9 = 20(Answer is 20)[Correct]

Results
Your name is: Jack
You played on: medium mode
You choosen this many questions: 5
you have got 3 / 5 correct
your scored : 60.0 /100
```

Figure 12Medium level output in custom game

C. If users' input is "hard"

```
else:
    (diff == "hard")
    print()
    customgame.mode3.hrad.hard()
```

So if the users input is hard it will execute the else statement and run customgame.mode3.hrad.hard()






| | | | |
|---|-------------|-------------------|-------------|
|  | __pycache__ | 4/3/2020 7:51 PM | File folder |
|  | customgame | 4/2/2020 12:08 AM | File folder |
|  | database | 4/4/2020 5:40 PM | File folder |
|  | quickgame | 4/3/2020 5:55 PM | File folder |
|  | main | 4/5/2020 6:08 PM | PY File |

Figure 13 All the folders



| | | | |
|---|-------|-------------------|-------------|
|  | mode1 | 3/31/2020 9:05 PM | File folder |
|  | mode2 | 3/31/2020 9:05 PM | File folder |
|  | mode3 | 3/31/2020 9:05 PM | File folder |

Figure 14 Inside the customgame package



| | | | |
|---|-------------|------------------|--------------|
|  | __pycache__ | 4/6/2020 7:58 PM | File folder |
|  | hard | 4/6/2020 7:58 PM | PY File 3 KB |

Figure 15 Inside mode3 where the hard level model is contained

Mode 3 contains the module for the hard difficulty which will execute when the user inserts his input

Hard difficulty python program:

```
def hard():
    import random
    #variables
    score=0
    rand=0
    update=[]
    #creating list with the operators in them
    op=["+", "-", "*", "/"]
    name=str(input("Enter name: "))
    times=int(input("How many questions would you like? = "))
    print()
    #main program
    for i in range(times):
        n1=random.randint(0,101)#importing 2 random values
        n2=random.randint(0,101)
        operators=random.choice(op)# used to choose a random value from the "op"
        #putting the whole question in a single line
        question=int(input(str(n1)+str(operators)+str(n2)+"="))
        #getting the real answer to see if random operator is "+"
        if(operators == "+"):
            ans=n1+n2
        #getting the real answer to see if random operator is "-"
        if(operators == "-"):
            ans=n1-n2
```

```

#getting the real answer to see if random operator is "*"
if(operators == "*"):
    ans=n1*n2
#checking if the answers match user input
if question == ans:
    #keeping track of the score
    score = score +1
    update.append(str(n1)+str(operators)+str(n2)+' = '+str(question)+'(Answer is '+str(ans)+')[Correct]')
else:
    update.append(str(n1)+str(operators)+str(n2)+' = '+str(question)+'(Answer is '+str(ans)+')[Incorrect]')

for val in range(times):
    print(update[val])

#printin the results
print()
print(" Results")
print("Your name is: ",name)
print("You played on: hard mode")
print("You choosen this many questions: ",times)
print("you have got",score,"/",times,"correct")
print("your scored : ",(score/times)*100,"/100")

```

Solution:

This python program is also inside of a function and then we import random inside the function. After that we created the variables that are the same as the variables in the medium difficulty python program but here in the op variable inside the list we have addition, subtraction and also multiplication, because this is the hard difficulty and we need more math operators for it.

For the main program we again use the for loop with “times” as range as because we need to run the program depending on the number of question that the user inputs. After that we again create two variables that will give us random numbers for that we use “range.randint” with range from 0 to 101 because we need to make the question difficult for the user, after that we make the “operators” variable as well so it will give us different math operators form the list in “op” variable and it can be done by using ”random.choice(op)”. Then I again create the question in a single line with the two random number and the random math operator given in the “op” variable and it is called the “question” variable.

After assigning the question we have to create the answers so that user can verify that there answer is correct or incorrect, we can do that by first getting the answers to all the operators and that be done by using an if condition for all the operators within the “op” variable, so if the variable “operators” is equal to addition then it will be $n1+n2$ and the answer will be assigned to the variable “ans” and we do that to all the math operators and all there answer of the two random number is saved to the variable “ans”. So, when we are finished getting all the answer on the possibility of getting one of the operators, we create an if condition and within the condition if “question” is equal to “ans” which is answer of one of those operators, If it is correct the score will get added by one and then we append the answer which will contain the two random number and the math operator and the user input and then it will show the real answer to the user and it will tell that the answer is correct, all this will be appended to the “update” variable which is a list, or if “question” is not equal “ans” it will repeat the same process but will tell the user that his answer is incorrect and show the real answer.

To print this update list, I created a for loop with range as “times” because it will print all the information in the list depending on the number question that is inputted by the user. Then it will print all the information in the update list.

Then for the results I again started by print the heading which is “results”, then the name, then the level of difficulty the user had played on which is hard, then the number of questions the user has chosen, then showing him how much he has scored out of the total question and finally showing him his score in percentage out of a hundred

Output:

```

Game menu
a) Quick Game
b) custom Game
c) Display Past Game Details
d) Exit

Enter your option:b

Custom Game

Enter difficulty (easy/medium/hard)= hard

Enter name: jess
How many questions would you like? = 5

69*3=207
39-76=-37
19-97=54
15+92=112
34*33=555
69*3 = 207(Answer is 207)[Correct]
39-76 = -37(Answer is -37)[Correct]
19-97 = 54(Answer is -78)[Incorrect]
15+92 = 112(Answer is 107)[Incorrect]
34*33 = 555(Answer is 1122)[Incorrect]

Results
Your name is: jess
You played on: hard mode
You choosen this many questions: 5
you have got 2 / 5 correct
your scored : 40.0 /100

```

Figure 16 Hard level output in custom game

2.4.3 If user input is “c”

```
elif (user_op == "c"):
    print()
    print ("  Past Game Detail")
    print()
    ask=str(input("View past score for Quick Game or Custom Game: "))
```

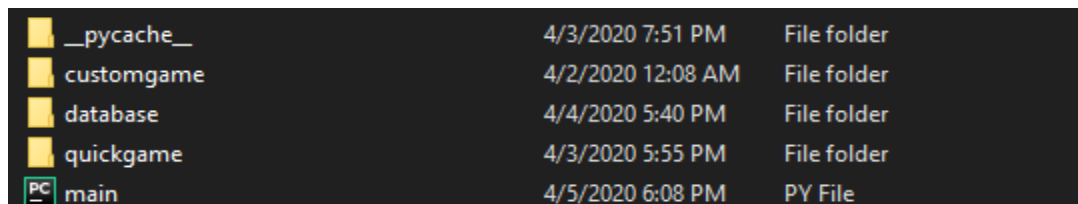
If user input is “c” it will execute the elif statement and will begin with printing the heading which is “past game results” then it will go to the “ask” variable which is there to ask if the user want to display the past game results for the quick game or custom game. For this I have created a package called “database” and inside it there are two modules one is called “cgamedatabase” which is for the custom game and “qgamedatabase” which is for the quick game

Then I created an if condition to check if the user would choose “Quick game” or else if he would choose “Custom game”

A. If user input “Quick game”

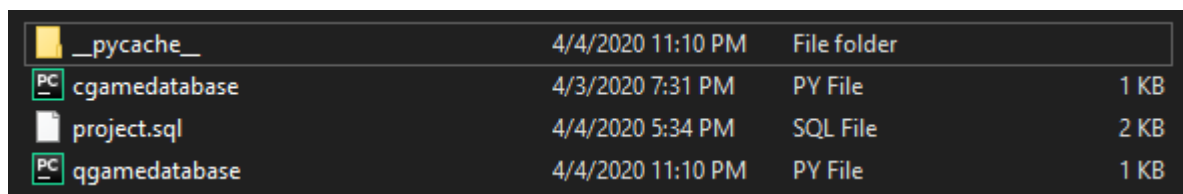
```
if(ask == "Quick game"):
    import database.qgamedatabase
```

If user input “Quick game” the program will execute database.qgamedatabase.



| | | |
|-------------|-------------------|-------------|
| __pycache__ | 4/3/2020 7:51 PM | File folder |
| customgame | 4/2/2020 12:08 AM | File folder |
| database | 4/4/2020 5:40 PM | File folder |
| quickgame | 4/3/2020 5:55 PM | File folder |
| main | 4/5/2020 6:08 PM | PY File |

Figure 17 Inside the database package



| | | | |
|---------------|-------------------|-------------|------|
| __pycache__ | 4/4/2020 11:10 PM | File folder | |
| cgamedatabase | 4/3/2020 7:31 PM | PY File | 1 KB |
| project.sql | 4/4/2020 5:34 PM | SQL File | 2 KB |
| qgamedatabase | 4/4/2020 11:10 PM | PY File | 1 KB |

Figure 18qgamedatabase module

Inside the database package we have both the modules for the results of customgame and quickgame.

Quick game database python program:

```
import mysql.connector

#open the connection with dictionary
conDict={"host":"localhost",
        "database":"project",
        "user":"root",
        "password":""}

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

#prepare a cursor object using cursor() method
cursor=db.cursor()

#execute sql query using execute() method
cursor.execute("SELECT * FROM quickgameresults")

#fetch results using fetchall() method
data=cursor.fetchall()

#printing the variables
print()
print("Name","Correct","Questions","Percentage")
for item in data:
    print("\r")
    for a in item:
        print(a,end="\t")

#closing the connection
db.close()
```

Solution:

We start by importing the connector first then we create a dictionary to store the connection and it will be called conDict and inside of that dictionary there will be the host, the database which is "project" which is the name of the database that the information is saved to, then the user which is "root" and finally the password which will be blank.

After that we store the connection to a variable that is called "db", then I create a cursor variable and assign that cursor to the "db" variable. Then I used that cursor variable and the "execute" command to write the SQL statement to show all the past results that are saved in the database and then in order to show the data that is saved in the database I created a variable called "data" and inside that variable I use the command "cursor.fetchall()" in order to bring the data that is stored in the database to the python program. Then I print the heading which are "name" users name, "correct" number of questions he got correct, "question" number of questions and "percentage" score in percentage and then create a for loop to print the data and finally I close the connection using the command "db.close()".

Now to save the results of the user from the quick game program, I created the program that connect to the database to store all the scores, names, number of questions and also the percentage of their score on the same program as the quick game.

Quick game python program with the database connection:

```
import mysql.connector

def quick():
    import random
#variables
    score=0
    rand=0
    update=[]
#asking the users name
    name=str(input("Enter name:"))
#main program
    for i in range(5):
        n1=random.randint(0,11)#importing 2 random values
        n2=random.randint(0,11)
#putting the whole question in a single line
        question=int(input(str(n1)+" "+str(n2)+"="))
#getting the real answer to see if it matches with the question
        ans=n1+n2
#checking if the answers match the useres input
        if question == ans:
            #keeping track of the score
            score = score +1
            update.append(str(n1)+' + '+str(n2)+' = '+str(question)+'(Answer is '+str(ans)+')[Correct]')
        else:
            update.append(str(n1)+' + '+str(n2)+' = '+str(question)+'(Answer is '+str(ans)+')[Incorrect]')

    for val in range(5):
        print(update[val])

#printing the results
    print(" Results")
    print("Your name is: ",name)
    print("you have got",score,"/5 correct")
    print("your scored : ",(score/5)*100,"/100")

#Saving the data to a database
    import mysql.connector
    #open the connection with dictionary
    conDict={"host":"localhost",
            "database":"project",
            "user":"root",
```

```

        "password":""}

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

#prepare a cursor object using cursor() method
cursor=db.cursor()

#execute sql query using execute() method
sqltxt="INSERT INTO quickgameresults (name,correct,totalquestions,percentage) Values
(%s,%s,%s,%s)"
userstxt=(name,score,"5",(score/5)*100)
cursor.execute(sqltxt,userstxt)
#inserting the change
db.commit()
#printing the connection
db.close()

```

Solution for the quick game program for storing the data to the database:

After the quick game program is executed the results will show the users name, his score and then the percentage of his score and that's the information we need to save so that the user can check his past game if he wants to.

In order to do that I first imported the connector and then I made a dictionary called "conDict" and we use the same connections that is inside of that dictionary that will be the host, the database which is "project" which is the name of the database that the information is saved to, then the user which is "root" and finally the password which will be blank. After that I again store the connection to a variable that is called "db", then we create a cursor variable and assign that cursor to the "db" variable, but here before I use the "cursor.execute" command I need to save the results to the database and I did that by creating a variable called "sqltxt" and inside that I use the SQL statement called "insert" to insert the results to the qgameresults table that is in the database, for the "insert" statement we need to first insert the order that qgameresults table is saved in that mean the attributes of the table, then the values that are going to be added is in the form "%s" because we want to save the results in the string format but we cannot input the results in this variable so I created another variable "userstxt" which contains the name, score, the number of questions and the score in percentage. This will automatically save the results no matter how many times the user plays the game in quick game. Now to save all this to the database we use the "cursor.execute(sqltxt,userstxt)" this will insert the data that is saved in "userstxt" to the database. But we must show that the data is actually been saved to the database so that where "db.commit()" which will print the execute statement (cursor.rowcount) which place all the data of the results in a row. Then we close the connection using "db.close()"

Output:

```
Game menu
a) Quick Game
b) custom Game
c) Display Past Game Details
d) Exit

Enter your option:c

Past Game Detail

View past score for Quick Game or Custom Game: Quick game

Name Correct Questions Percentage
nem      1         5         20
nem      2         5         40
```

Figure 19 Output of the past game results from Quick game

B. If users' input is "Custom game"

else:

(ask == "Custom game")

import database.cgamedatabase

If user input is "Quick game" then it will run else statement which will execute database.cgamersults.






| | | | |
|---|-------------|-------------------|-------------|
|  | __pycache__ | 4/3/2020 7:51 PM | File folder |
|  | customgame | 4/2/2020 12:08 AM | File folder |
|  | database | 4/4/2020 5:40 PM | File folder |
|  | quickgame | 4/3/2020 5:55 PM | File folder |
|  | main | 4/5/2020 6:08 PM | PY File |

Figure 20 All folders





| | | | | |
|---|---------------|-------------------|-------------|------|
|  | __pycache__ | 4/4/2020 11:10 PM | File folder | |
|  | cgamedatabase | 4/3/2020 7:31 PM | PY File | 1 KB |
|  | project.sql | 4/4/2020 5:34 PM | SQL File | 2 KB |
|  | qgamedatabase | 4/4/2020 11:10 PM | PY File | 1 KB |

Figure 21 Inside the database package

Custom game database python program:

```
import mysql.connector
    #open the connection with dictionary
conDict={"host":"localhost",
        "database":"project",
        "user":"root",
        "password":""}

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

#prepare a cursor object using cursor() method
cursor=db.cursor()

#execute sql query using execute() method
cursor.execute("SELECT * FROM customgameresults")

#fetch results using fetchall() method
data=cursor.fetchall()

#printing the variables
print()
print("Name","Difficulty","Correct","Questions","Percentage")
for item in data:
    print("\r")
    for a in item:
        print(a,end="\t")

#closing the connection
db.close()
```

Solution:

We again start by importing the connector first then we create a dictionary to store the connection and it will be called conDict and inside of that dictionary there will be the host, the database which is "project" which is the name of the database that the information is saved to, then the user which is "root" and finally the password which will be blank.

After that we store the connection to a variable that is called "db", then I create a cursor variable and assign that cursor to the "db" variable. Then I used that cursor variable and the "execute" command to write the SQL statement to show all the past results that are saved in the database and then in order to show the data that is saved in the database I created a variable called "data" and inside that variable I use the command "cursor.fetchall()" in order to bring the data that is stored in the database to the python program. Then I print the heading which are "name" users name, "difficulty" which difficulty the user had played on, "correct" number of questions he got correct, "question" number of questions and "percentage" score in percentage a for loop to print the data and final I close the connection using the command "db.close()".

Now to save the results of the user, I created the program that connect to the database to store all the scores, names, number of questions, percentage of their score and the difficulty they played on, on the same program as the difficulty levels are kept in which is mode1, mode2 and mode3.

Model “easy” difficulty python program:

```
def ez():
    import random
    #variables
    score=0
    rand=0
    update=[]
    name=str(input("Enter name: "))
    times=int(input("How many questions would you like? = "))
    print()
    #main program
    for i in range(times):
        n1=random.randint(0,11)#importing 2 random values
        n2=random.randint(0,11)
        #putting the whole question in a single line
        question=int(input(str(n1)+"+"+str(n2)+"="))
        #getting the real answer to see if it matches with the question
        ans=n1+n2
        #checking if the answers match user input
        if question == ans:
            #keeping track of the score
            score = score +1
            update.append(str(n1)+' + '+str(n2)+' = '+str(question)+'(Answer is '+str(ans)+')[Correct]')
        else:
            update.append(str(n1)+' + '+str(n2)+' = '+str(question)+'(Answer is '+str(ans)+')[Incorrect]')

    for val in range(times):
        print(update[val])

#printing the results
print()
print(" Results")
print("Your name is: ",name)
print("You played on: easy mode")
print("You choosen this many questions: ",times)
print("you have got",score,"/",times,"correct")
print("your scored : ",(score/times)*100,"/100")

#saving the data to a database
import mysql.connector
#open the connection with dictionary
```

```

conDict={"host":"localhost",
        "database":"project",
        "user":"root",
        "password":""}

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

#prepare a cursor object using cursor() method
cursor=db.cursor()

#execute sql query using execute() method
sqltxt="INSERT INTO quickgameresults (name,correct,totalquestions,percentage) Values
(%s,%s,%s,%s)"
utxt=(name,"easy",score,times,(score/5)*100)
cursor.execute(sqltxt,utxt)
#inserting the change
db.commit()
#closing the connection
db.close()
return

```

Mode2 “medium” difficulty python program:

```

def med():
    import random
    #variables
    score=0
    rand=0
    update=[]
    #creating list with the operators in them
    op=["+", "-"]
    name=str(input("Enter name: "))
    times=int(input("How many questions would you like? = "))
    print()
    #main program
    for i in range(times):
        n1=random.randint(0,51)#importing 2 random values
        n2=random.randint(0,51)
        operators=random.choice(op)# used to choose a random value from the "op"
        #putting the whole question in a single line
        question=int(input(str(n1)+str(operators)+str(n2)+"="))
        #getting the real answer to see if random operator is "+"
        if(operators == "+"):
            ans=n1+n2
        #getting the real answer to see if random operator is "-"
        if(operators == "-"):
            ans=n1-n2
        #checking if the answers match user input
        if question == ans:
            #keeping track of the score

```

```

        score = score +1
        update.append(str(n1)+str(operators)+str(n2)+' = '+str(question)+'(Answer is
'+str(ans)+')[Correct]')
    else:
        update.append(str(n1)+str(operators)+str(n2)+' = '+str(question)+'(Answer is
'+str(ans)+')[Incorrect]')

```

```

for val in range(times):
    print(update[val])

```

#printin the results

```

print()
print(" Results")
print("Your name is: ",name)
print("You played on: medium mode")
print("You choosen this many questions: ",times)
print("you have got",score,"/",times,"correct")
print("your scored : ",(score/times)*100,"/100")

```

#saving the data to a database

```

import mysql.connector
#open the connection with dictionary
conDict={"host":"localhost",
        "database":"project",
        "user":"root",
        "password":""}

```

```

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

```

#prepare a cursor object using cursor() methord

```

cursor=db.cursor()

```

#execute sql query usinng execute() methord

```

sqltxt="INSERT INTO qickgameresults (name,difficulty,correct,totalquestions,percentage)
Values (%s,%s,%s,%s,%s)

```

```

utxt=(name,"medium",score,times,(score/5)*100)
cursor.execute(sqltxt,utxt)

```

#inserting the change

```

db.commit()

```

#closing the connection

```

db.close()
return

```

Mode3 “hard” difficulty python program:

```
def hard():
    import random
    #variables
    score=0
    rand=0
    update=[]
    #creating list with the operators in them
    op=["+", "-", "*",]
    name=str(input("Enter name: "))
    times=int(input("How many questions would you like? = "))
    print()
    #main program
    for i in range(times):
        n1=random.randint(0,101)#importing 2 random values
        n2=random.randint(0,101)
        operators=random.choice(op)# used to chooose a random value from the "op"
        #putting the whole question in a single line
        question=int(input(str(n1)+str(operators)+str(n2)+"="))
        #getting the real answer to see if random operator is "+"
        if(operators == "+"):
            ans=n1+n2

        #getting the real answer to see if random operator is "-"
        if(operators == "-"):
            ans=n1-n2
        #getting the real answer to see if random operator is "*"
        if(operators == "*"):
            ans=n1*n2
        #checking if the answers match user input
        if question == ans:
            #keeping track of the score
            score = score +1
            update.append(str(n1)+str(operators)+str(n2)+' = '+str(question)+'(Answer is '+str(ans)+')[Correct]')
        else:
            update.append(str(n1)+str(operators)+str(n2)+' = '+str(question)+'(Answer is '+str(ans)+')[Incorrect]')

    for val in range(times):
        print(update[val])

#printin the results
print()
print("  Results")
print("Your name is: ",name)
print("You played on: hard mode")
print("You choosen this many questions: ",times)
```



```

print("you have got",score,"/",times,"correct")
print("your scored : ",(score/times)*100,"/100")

#saving the data to a database
import mysql.connector
#open the connection with dictionary
conDict={"host":"localhost",
        "database":"project",
        "user":"root",
        "password":""}

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

#prepare a cursor object using cursor() method
cursor=db.cursor()

#execute sql query using execute() method
sqltxt="INSERT INTO quickgameresults (name,difficulty,correct,totalquestions,percentage)
Values (%s,%s,%s,%s,%s)
        utxt=(name,"hard",score,times,(score/5)*100)
        cursor.execute(sqltxt,utxt)
#inserting the change
        db.commit()
#closing the connection
db.close()

```

Solution for the custom game program for storing the data to the database:

So after one of the difficulty levels in the custom game package is executed the results will show the users name, which difficulty he played on ,his score, total number of questions and then the percentage of his score and that's the information needs to be saved so that the user can check his past game if he wants to.

In order to do that I again imported the connector and then I made a dictionary called "conDict" and we use the same connections that is inside of that dictionary that will be the host, the database which is "project" which is the name of the database that the information is saved to, then the user which is "root" and finally the password which will be blank. After that I again store the connection to a variable that is called "db", then we again create a cursor variable and assign that cursor to the "db" variable, but here before I use the "cursor.execute" command I need to save the results to the database and I did that by creating a variable called "sqltxt" and inside that I use the SQL statement called "insert" to insert the results to the cgameresults table that is in the database, for the "insert" statement we need to first insert the order that cgameresults table is saved in that means the attributes of the table, then the values that are going to be added is in the form "%s" because we want to save the results in the string format but we cannot input the results in this variable so I created another variable "userstext" which

contains the name, difficulty of the level, score, the number of questions and the score in percentage. This will automatically save the results no matter how many times the user plays the game in custom game. Now to save all this to the database we use the same statement “cursor.execute(sqltxt,userstxt)” this will insert the data that is saved in “userstxt” to the database. But we must show that the data is actually been saved to the database so that where “db.commit()” which will print the execute statement (cursor.rowcount) which place all the data of the results in a row. Then we have to close the connection using “db.close()”

output:

```
Game menu
a) Quick Game
b) custom Game
c) Display Past Game Details
d) Exit

Enter your option:c

Past Game Detail

View past score for Quick Game or Custom Game: Custom game

Name Difficuiltly Correct Questions Percentage
Nemsara easy 4 5 80
Nmesara medium 0 1 0
Jack medium 3 5 60
jess hard 2 5 40
Nem medium 4 5 80
```

Figure 22 Output of the past game results from Custom game

2.4.4 If user input is “d”

```
elif(user_op == "d"):
    exit()
```

If users’ input is “d” it will execute the command which is “exit()” and will end the program.

output:

```
Game menu
a) Quick Game
b) custom Game
c) Display Past Game Details
d) Exit

Enter your option:d

C:\Users\nemsa\Documents\P2realproject>_
```

3. Test cases

| Test Case ID | Test Scenario | Test Steps | Test Data | Expected | Actual Results Pass/Fail |
|--------------|---------------|---|-------------------------------|--|-----------------------------|
| Test 1 | Main menu | 1.Creating the game menu inside of the Function. 2.To run the menu use “while true” loop for call functions. 3.inside the menu each selection print Statements. | Enter Option= 1, 2, 3, 4,5 | user enter ‘1’=print statement ‘1’ user enter ‘2’=print statement ‘2’ user enter ‘3’=print statement ‘3’ user enter ‘4’=print statement ‘4’ user enter ‘5’=print statement ‘5’ | pass |
| Test 2 | Game menu | 1.Creating the game menu inside of the Function. 2.To run the menu used print statements that the options are given. 3.inside the menu each selection print Statements. | Enter Option= a, b, c, d | user enter ‘a’=print statement ‘a’ user enter ‘b’=print statement ‘b’ user enter ‘c’=print statement ‘c’ user enter ‘d’=print statement ‘d’ user enter ‘e’=print statement ‘e’ | pass |
| Test 3 | Quick game | 1.User inputs the name. 2. user inputs answer | Name=Jack Answer=54 | If user inputs: Jack = print statement ‘jack’ If answer is: 54=print statement 54 | pass |
| Test 4 | Quick game | 1.User inputs the name. 2. user inputs answer | Name=Jack Answer=54 | If user inputs: Jack = print statement ‘jack’ If answer is: Jack=error | fail |

| Test Case ID | Test Scenario | Test Steps | Test Data | Expected | Actual Results Pass/Fail |
|--------------|-----------------------------|---|---|---|-----------------------------|
| Test 5 | Custom game Easy level | 1.user input the name 2.user input answers | name=Jack answer=54 | If user inputs: Jack = print statement 'jack' If answer is: 54=print statement 54 | pass |
| Test 6 | Custom game Easy level | 1.user input the name 2.user input answers | name=Jack answer=54 | If user inputs: Jack = print statement 'jack' If answer is: Jack=error | fail |
| Test 7 | Custom game Medium Level | 1.user input the name 2.user input answers | Name=Jack Answer=54 | If user inputs: Jack = print statement 'jack' If answer is: 54=print statement 54 | pass |
| Test 8 | Custom game Medium level | 1.user input the name 2.user input answers | Name=Jack Answer=54 | If user inputs: Jack = print statement 'jack' If answer is: Jack=error | fail |
| Test 9 | Custom game Hard level | 1.user input the name 2.user input answers | Name=Jack Answer=54 | If user inputs: Jack = print statement 'jack' If answer is: 54=print statement 54 | pass |
| Test 10 | Custom game Hard level | 1.user input the name 2.user input answers | Name=Jack Answer=54 | If user inputs: Jack = print statement 'jack' If answer is: Jack=error | fail |
| Test 11 | Past game results detail | 1.Creating the game menu inside of the Function. 2.To run the menu use "while true" loop for call functions. 3.inside the menu each selection print Statements. | Enter option= Quick game or Custom game | If user inputs quick game: Quick game=print the results of quick game. If user inputs custom game: Custom game=print the results of custom game. | pass |

4. Conclusion

So conclusion this program may look easy to construct but took effort to make, with trial and errors to get it working properly and also the time spent to construct such a program took a few days to do so and constant testing of different method's to ensure that there is not a single error that would occur when running this program.