**Python Programming Assignment – Documentation**

**Adam Chell – N0815742**

**Program Description**

The program that I am going to design and create using the python language and IDLE application will be quiz based with many options for customisation. The basic quiz components will check the inputted answer by the user against the stored answer within the program and either output correct and add one to the user’s score, or output incorrect and keep the user’s score the same. It will allow the user to choose multiple factors such as the topic of the quiz, the length of the quiz and even create their own quiz. These questions and answers will be read from external text files that contain the different topics. If the user wants, they will be able to create and save their own quiz within the program that can then be chosen at the beginning. The user will answer each question one at a time and then their total score will be displayed at the end with how many they got correct. The user will then be taken back to the main menu to choose another topic or they can choose to restart the quiz.

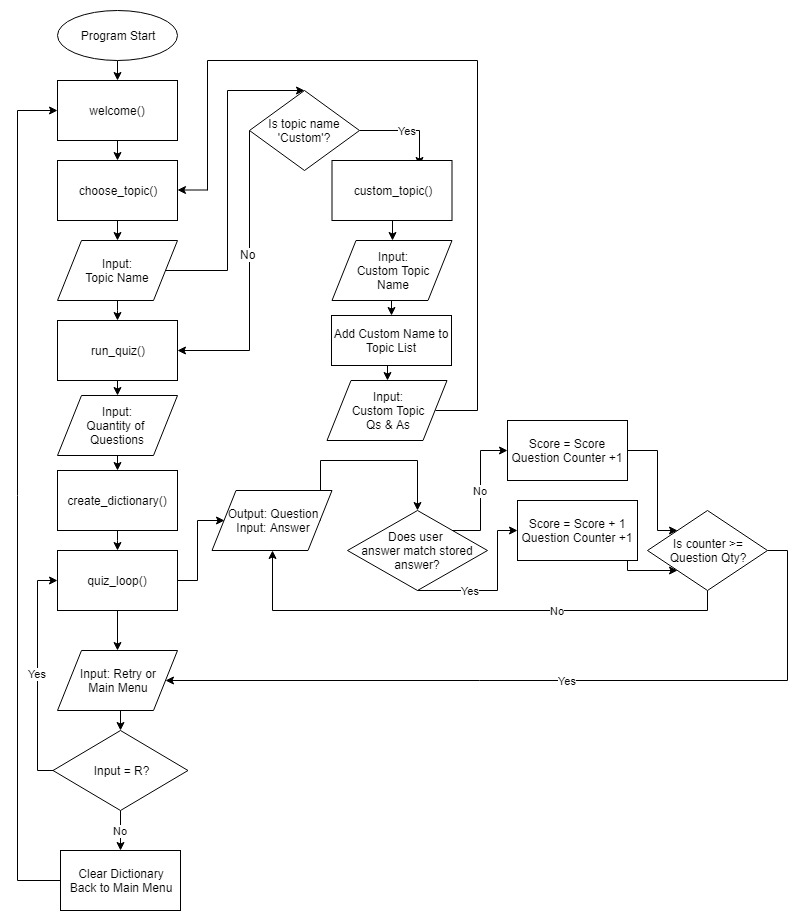
**Analysis of Requirements of Program**

The program must be able to:

* Output questions to the user that are read from an external file that contains all of the questions and answers for that specific topic. This will be done by creating a .txt file for each of the quiz topics which lists the questions and answers that are then opened within the program and converted and stored into key: value pairs in a dictionary. This will be achieved through file handling within the program.
* Check if the answer input by the user matches the value of the answer stored within the topic’s text file and therefore the dictionary within the program. This will be done by using loops and if statements to check the string input against the stored value and the result will be output to the user to say if they got the question correct or incorrect.
* Allow the user to choose the topic in which they want to answer questions on.

This could be done by printing each of the available topics from a list and receiving an input from the user that matches one of the options that will then run that quiz.

* Allow the user to choose the length of the quiz, by letting them input how many questions they want to answer. This will be achieved by creating a loop with a counter that outputs the questions. The counter will count up to the value the user has input and then break the loop once it has been reached.
* Calculate how many questions the user has got correct at the end of the quiz and then output this to the user. This will be done by adding a counter into the question loop that keeps track of score and stores it in a variable. This will then be output along with the number amount of questions being asked to the user so that they know how many they got right out of a possible maximum.
* Allow the user to create their own quiz with their own questions and answers that can be then ran from the main menu. This will be done by creating a topic called Custom that can be entered and then it will ask the user to input a topic name of their choice and then they input the questions one by one. This will be then written to a .txt file to be read from later and converted into a dictionary in the same way as the other topics.
* Allow the user at the end of each quiz to either retry the same quiz or to return back to the main menu. This will be achieved by creating an input with an if statement that simply restarts the quiz loop or restarts the program as a whole but keeps the variables and stored data.

**Flowchart of Program**

**Functions**

welcome()

* Prints a welcome message.
* Calls choose\_topic() function

choose\_topic()

* Prints question asking which topic the user would like to be asked questions on.
* Prints the topics out one by one from the topic list.
* If statements for each of the topic names and one for custom topic. If value is entered the variable that holds the topic name and the name of the text file is updated.
* Prints an error message if none of the values match

custom\_topic()

* Prints out a message asking for a custom topic name and then enters that input into the topic names list.
* Opens the custom text file to be written to.
* Runs a while loop that runs until 10 questions and answers have been input.
* Strips the inputted text of new lines and also inserts a colon between the question and answer.
* Saves the updated file.

run\_quiz()

* Prints a message that welcomes the user to the name of the quiz.
* Asks the user how many questions they would like to answer and stores it in a variable.
* Calls the create\_dictionary() and then the quiz\_loop() functions.

create\_dictionary()

* Uses the variable that stores the chosen topic’s text file to open it up and convert it into a dictionary that is stored within the program to be used in the quiz.
* For each line in the text file the newline is stripped and then split where the colon appears into key: value pairs in the dictionary.
* Updates the dictionary to the current chosen topic’s questions.

quiz\_loop()

* Outputs each question from the dictionary by printing the key and then asking for an input for the answer.
* If statements are used to compare the inputted answer against the stored value in the dictionary. If the answer is correct, a message is output and the score is increased.
* The loop runs until the question amount reaches the number chosen by the user that is stored in the variable.
* Outputs the users score at the end as well as the total possible score so the user can see how well they did.
* The user can choose to restart the quiz or return to the main menu by inputting either R or M. If R is chosen, quiz\_loop() is restarted but if M is chosen the dictionary is cleared and the welcome() function is called to restart the program.

**Testing the Program’s Main Features**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Description | Type of Data | Data Input | Expected Outcome | Actual Outcome | Pass/Fail? |
| Choosing a quiz topic. | Normal | Animals | Program will load the animals quiz. | The program successfully loads animals quiz. | Pass |
| Choosing a quiz topic. | Extreme | animals | Program will load the animals quiz. | The program successfully loads animals quiz. | Pass |
| Choosing a quiz topic. | Erroneous | animal | Program will not load the animals quiz and tell the user to choose a valid option. | The program output a message saying choose a valid option. | Pass |
| Choosing the custom quiz topic. | Normal | Custom | Program will run the custom quiz creation and ask the user for a custom topic name. | The program then asked for the user to enter a custom topic name. | Pass |
| Choosing the custom quiz topic. | Extreme | CuStOm | Program will run the custom quiz creation and ask the user for a custom topic name. | The program then asked for the user to enter a custom topic name. | Pass |
| Choosing the custom quiz topic. | Erroneous | Customm | Program will not run the custom quiz creation and ask for a valid option to be chosen. | Program did not run the custom quiz creation and ask for a valid option to be chosen. | Pass |
| Choosing how many questions to be asked. | Normal | 6 | The program will output only 6 questions to the user and then end the quiz. | The program output only 6 questions to the user and then end the quiz. | Pass |
| Choosing how many questions to be asked. | Extreme | 10 | The program will output only 10 questions to the user and then end the quiz. | The program output only 10 questions to the user and then end the quiz. | Pass |
| Choosing how many questions to be asked. | Erroneous | 11 | The program will ask the user to input a number between 1 and 10. | The program asked the user to input a number between 1 and 10. | Pass |
| Answering a question in the quiz. | Normal | Cheetah | The program will output ‘Correct’ and then move onto the next question. | The program output ‘Correct’ and then moved onto the next question. | Pass |
| Answering a question in the quiz. | Extreme | cHeeTaH | The program will output ‘Correct’ and then move onto the next question. | The program output ‘Correct’ and then moved onto the next question. | Pass |
| Answering a question in the quiz. | Erroneous | Cheeta | The program will output ‘Incorrect’ and then move onto the next question. | The program output ‘Incorrect’ and then moved onto the next question. | Pass |
| Choosing whether to restart the same quiz or return to the main menu. | Normal | R | The program will restart the quiz with the same topic and questions. | The program restarted the quiz with the same topic and questions being output. | Pass |
| Choosing whether to restart the same quiz or return to the main menu. | Normal | M | The program will return to the main menu and output the welcome message. | The program returned to the main menu and output the welcome message. | Pass |
| Choosing whether to restart the same quiz or return to the main menu. | Erroneous | Main Menu | The program will output an error message saying please choose a valid option. | The program output an error message saying please choose a valid option. | Pass |

**What went well and what could be improved:**

Overall, I am extremely pleased with how well the program functions as I feel it has met all of the criteria that I set out before I started writing it. I did not have to remove or change any of the features due to them being too complex or time consuming to figure out.

The file handling that is part of the program works without flaws every time due to the strict way in which the text within the file is formatted and read. The text files were converted into dictionaries that were then stored within the program so that they could be accessed to compare against the user’s own input and also to output the questions to the user.

Another part of the program that I feel works really well is how it actually appears visually to the user. This is done by making sure there is a consistent layout throughout the whole quiz and also by using the sleep function to add a timer to certain pieces of code so that there is a delay between them being printed, so that the user has time to read each line rather than a massive block being printed each time.

Although I have met all of the program’s requirements, upon reflection it could still be improved by changing some small aspects of the code that make the program even more efficient.

One improvement to the code could be that instead of the custom.txt being opened and written to, and then read from to create the custom dictionary, instead the program could write a new file with the name being the custom topic name and then this topic gets permanently added to the list of topics even after the program is closed, as currently the custom topic resets after the program is closed down. Although this is not essential, it provides a more interesting experience as there will be many custom quizzes available from the start.

One feature that I would have liked to add would be at the end of each quiz, there would be an option for the user to see all of the correct answers to the questions they answered incorrectly. This would be done by the program remembering which they got incorrect and then adding the correct answer to a list to then be output all together.