Task 1 - Quiz

# 

# 

# Starting Menu

When the user first starts the program they are presented with the ‘Starting Menu’. This lists the three options – ‘1) Log In’, ‘2) Create New Account’, and ‘3) Generate Report’ – each indexed so the user can easily select their desired option and so that the input can easily be verified as either 1, 2, or 3. This uses a text based interface because the requirements do not specify he requirement for a graphical interface.

# Log In

If ‘Log In’ is selected, the user will be prompted to enter their username. The program will validate that the user has not entered ‘top’, since that is used by the program as described later in this section. The program will then open the ‘user\_info.json’ file (using a ‘try except’ statement in case the file does not exist) into a master ‘users’ dictionary where the username is the key. It will then search the list for the user information and load it into the ‘user’ variable (if an account is present). I will use a dictionary to store all the user information so that information can be requested with just a username rather that by looping through a list. I will be using a structured ‘.json’ file so that the program can easily store and read lists and dictionaries (the data types that the user info is stored in). The program will also load the ‘top’ dictionary within the ‘users’ dictionary, that stores the top score achieved for each topic and the username of the achiever.

If the user account is present, then the program will prompt the user to enter a password, checking against the user information that it is correct. If the password is valid, then the program will move on to the ‘Topic Menu’.

# Generate Report

This Menu is similar to the ‘Starting Menu’ but gives the user two options – ‘1) User Report’, and ‘2) Topic Report’.

## User Report

This section will load the ‘user\_info.json’ file and request the user to input the username of the student they want to retrieve information from (verifying if the user exists). The program will then print out the user information and then the quiz information (topic, score, and grade) each on a new line.

## Topic Report

This section will load the ‘user\_info.json’ file and present the user with a menu similar to the ‘Starting Menu’ but the three options will be: ‘1) History’, ‘2) Music’, and ‘3) Computer Science’. The user will then select the difficulty of the topic with a menu similar to the ‘Staring Menu’ with the three choices being: ‘1) Easy’, ‘2) Medium’, and ‘3) Hard’. The program will then load the information of each student storing their results in a temporary list. The program will then print the topic name, difficulty, average score, highest score(s) achieved and their user information.

# Create New Account

This section will create the necessary variables for the new user’s account. The ‘user’ variable will be laid out like this: user = {‘user’ : ‘*username*’, ‘pass’ : ‘*password*’, ’name’ : ‘*name*’, ‘age’ : *age*, ‘year’ : ‘*year group*’, ‘quizzes’ : [*list of the quizzes taken (explained in ‘Quiz Finish’ section)*]}. The user information is stored in a dictionary so that I can easily code the program without having to remember the list index for each section of user data.

This section will also open the ‘user\_info.json’ file storing all the user information in a master ‘users’ list and loading in the information described in the ‘Log In’ section.

# Create Username

This section will first prompt the user to enter in their full name – validating that their name only contains no numbers, and a space (to make sure that it is a full name). The program will then prompt the user for their age – validating that it is an integer. The program will then prompt the user to enter their year group – once again validating that it is an integer. The name and age will be used to generate the username – with the first three letters of their name and their age. This will be checked to make sure that it does not already exist, and all the personal information will be stored in the ‘user’ dictionary. I have chosen for the user to enter their year group now, even though it is not needed to create a username so that all the personal information can be entered at once.

This section will then print the users username so that they can note it down.

# Create Password

This section will prompt the user to enter in their desired password (reminding them that it is case sensitive), asking them to confirm their password by typing it again so that the user doesn’t accidently enter in an undesired password.

The program will then print out both the username and password of the newly created account so they can note it down.

# Topic Menu

This section will first prompt the user, using the same menu system that ‘Starting Menu’ uses but prompting the user to choose form the three topics (history, music, and computer science).

The section will then load the questions and answers from the ‘q\_and\_a\_music.txt’ file if music has been selected. The file will be laid out where the first line of the file is ignored (this will tell the user how to edit the questions contained within the file). The next line will contain the question, and the next will be the correct answer, and the next three will be incorrect answers. This pattern will continue for each question. The questions and answers are contained on different lines so that the teacher can user punctuation such as ‘,’ in the questions and answers and not cause the program to malfunction. The questions and answers are stored in a ‘.txt’ file so that the teacher can easily edit the questions and answers without having to know how to use a structured file such as a ‘.json’.

# Difficulty Menu

This section will prompt the user to elect a difficulty (easy, medium, and hard) using the same menu system that ‘Starting Menu’ uses. Storing the users answer as a ‘2’ for easy, ‘3’ for medium, and ‘4’ for hard. This corresponds to the number of answers each difficulty presents to the user. This section will also create a ‘score’ variable for tracking the user’s score.

# Question

This section will loop as many times as there are questions in a topic, so the teacher can add as many questions as they want.

The section will first print out the question loaded in the ‘Topic Menu’ Section. It will then generate a number between 1 and the number of answers being printed out (2, 3, or 4) and store it. This will act as an indicator to when to put the correct answer. As the program prints out the possible answers, it will check if the answer number it generated earlier. If so, then it will print out the correct answer; and if not, it will print out an incorrect answer. This is so that the correct answer is not always printed first, making it easy to chat at the quiz. The program will then use a menu indexed with letters (‘A’, ‘B’, ‘C’, and ‘D’) so the user can select which answer they think is correct. The program will verify that the user has only entered a valid option (e.g. If they enter ‘D’ on an easy rated quiz).

If the user has answered correctly the score variable will increase by 1 and a message will be printed saying they have answered correctly. If not, then the program will say that the user has answered incorrectly, telling them the correct answer.

## Quiz Finish

After all he sections have been completed, the program will generate the grade that the user has achieved. The grades will be as follows: 10% = 1, 20% = 2, 30% = 3, 40% = 4, 50% = 5, 60% = 6, 70% = 7, 80% = 8, and 90% = 9. This grade will be printed along with the percentage and the score that the user has achieved. The grade and percentage of the student will not be stored so as to save memory and will be generated on demand using their own functions. The information about the quiz will be stored in a dictionary within the ‘quizzes’ list within the ‘user’ dictionary. The formatting of the dictionary will be as follows: {‘topic’ : ‘*topic*’, ‘difficulty’ : ‘*difficulty*’, ‘score’ : ‘*score*’, ‘totalQs’ : ‘*total number of questions asked*’}. The total number of questions asked is stored in here so that the number of questions can be changed and an accurate percentage and grade can still be calculated. The program will also check if the score is higher than or equal to that stored in the dictionary ‘top’ described earlier. If so, then the program will replace the ‘score’ number in the dictionary with that achieved by the latest user and will also replace the username with that of the correct achiever.

Once this information has been stored, the ‘user’ dictionary will be added to the ‘users’ master dictionary and written to the ‘user\_info.json’ file.

# Success Criteria

* The user will be presented with three options: ‘Log In’, ‘Create New Account’, and ‘Generate Reports’.
* The input will be validated so that they can only select these three options.
* **Generate Reports:**
  + The user will be presented with two options: ‘User Report’ and ‘Topic Report’.
  + The input will be validated so that they can only select these two options.
  + **User Report:**
    - The user will be able to enter the username of an account.
    - The input will be validated so that they can only enter the username of an existing account.
    - The program will print out all of the quizzes the user has taken and their grade for each one.
  + **Topic Report:**
    - The user will be presented with three options: ‘History’, ‘Music’, and ‘Computer Science’.
    - The input will be validated so that they can only select these three options.
    - The program will then print out the user information and then the quiz information (topic, score, and grade) each on a new line.
* **Log In:**
  + The user will be able to enter their username.
  + The input will be validated to make sure that the user exists and is not ‘top’.
  + The user will be able to enter their password.
  + The input will be checked against the password stored in the ‘user’ dictionary.
* **Create New Account:**
  + The user will be able to enter their full name.
  + The input will be validated to make sure that it contains no numbers and is a full name (containing a space).
  + The user will be able to enter their age and year group.
  + These inputs will be validated to make sure that they are integers.
  + The program will generate a username off this information.
  + The user will be able to enter their desired password.
  + All of this information will be stored in a user dictionary.
* The user will be presented with three topics to choose from: ‘History’, ‘Music’, and ‘Computer Science’.
* The input will be validated so that they can only select these three options.
* The user will be presented with three difficulties to choose from: ‘Easy’, ‘Medium’, and ‘Hard’.
* The input will be validated so that they can only select these three options.
* The questions and answers will be loaded from an external file.
* The question will be printed out first with the answers then following on the next few lines.
* Each answers will be prefaced with ‘A)’, ‘B)’, ‘C)’, or ‘D’.
* The amount of answers will be different for each difficulty: ‘2’ for easy, ‘3’ for medium, and ‘4’ for hard.
* The correct answer will be randomly placed within the 2 – 4 answers print out, but will be present.
* The user will be able to select one of the four answers using the letter preceding them.
* The input will be validated so they can only choose the answers present (e.g. not being able to choose ‘D’ on easy difficulty.
* If the answer the user selects is correct, then a congratulation message will be printed and their score will increase by +1.
* If the answer the user selects is incorrect, then they will be told that the correct answer was.
* The question section will loop for as many questions there are in the topic.
* One all the questions have be answered, then the score, percentage, and grade will be printed.
* This information will be stored in the ‘quizzes’ list in the ‘user’ dictionary.
* The ‘user’ dictionary will be stored in the ‘users’ dictionary.
* The ‘users’ dictionary will be stored in the ‘user\_info.json’ file.

# Testing

# Starting Menu

In this subprogram, the user input will be tested to see that they have only entered one of the possible choices (‘A’, ‘B’, or ‘C’) the case of the letter will not matter as the program will automatically convert the input to upper case – so that the user can enter in ‘A’ or ‘a’ and get the same result without there needing to be a statement in the code that individually checks for ‘A’ and ‘a’. If the input was not validated, the program would not crash, but the program would not run any further, decreasing its functionality.

# Log In

This subprogram will first need to check that the ‘user\_info.json’ file is present (through a ‘try except’ statement), if the program dod not do this and the file was not there than the program would throw a IOError. The program would then need to validate that the user’s inputted username matches one on the file. It will do this by trying to access the dictionary key of that username within a ‘try except’ statement. If the program throws an error within this statement, then the program will request the user to enter a valid username. The program will also check that the user has not entered ‘top’ as that is where the high scores are stored.

The subprogram will then need to verify that the password that the user has entered matches the one stored in the ‘user’ variable. This is so a user’s account cannot be accessed without knowing their password.

# Generate Report

This subprogram will validate in the same way as he ‘Starting Menu’ subprogram but will only have two options (‘A’ and ‘B’)

# User Report

This subprogram will need to verify that the username entered is valid. It will do this by trying to access the dictionary key of that username within a ‘try except’ statement. If the program throws an error within this statement, then the program will request the user to enter a valid username. This subprogram will also validate the existence of the ‘user\_info.json’ file in the same way that the ‘Log In’ subprogram does.

# Topic Report

This subprogram will validate the input of the user in the same way that the ‘Starting Menu’ subprogram does. This subprogram will also validate the existence of the ‘user\_info.json’ file in the same way that the ‘Log In’ subprogram does.

# Create New User

This subprogram will validate the existence of the ‘user\_info.json’ file in the same way that the ‘Log In’ subprogram does.

# Create Username

This subprogram will first validate that the user has entered a valid name. This is defined as containing no numbers (this will be done by taking each index of the entered string an trying to convert it into an integer type within a ‘try except’ statement), and containing one space so as to make sure that the user has entered their full name (this will be done by going through each index of the string and checking that at least one of them is a space).

The subprogram will then validate that the user has entered a valid age and year group. It will do this by converting it into an integer type within a ‘try except’ statement and if an exception is thrown then the user will be required to enter a valid age or year group.

# Create Password

This subprogram will not need to validate the password the user enters, as there are no requirements in the scenario that state limiting factors on the user’s password.

# Topic Report

This subprogram will validate the input of the user in the same way that the ‘Starting Menu’ subprogram does.

# Difficulty Menu

This subprogram will validate the input of the user in the same way that the ‘Starting Menu’ subprogram does.

# Question

This subprogram will validate the input of the user in the same way that the ‘Starting Menu’ subprogram does, but will change the amount of valid options for the user’s input depending on the difficulty set in the ‘Difficulty Menu’ (2 for ‘Easy’, 3 for ‘Medium’, and 4 for ‘Hard’) and will use letters for indexing rather than numbers.

# User Interface

# Menu

1. Log In
2. Create New Account
3. Generate Reports

>>> *[USER INPUT]*

This is the main user menu and will change depending on what menu the user is accessing (‘Starting Menu’, ‘Generate Report’, ‘Topic Report’, ‘Topic Menu’ or ‘Difficulty Menu’). This example is of the ‘Starting Menu’ interface.

That is not a valid option. Please try again.

>>> *[USER INPUT]*

This will be presented to the user if they enter an invalid menu option. I have decided not to reprint the menu options so as not to clutter the screen.

# Account creation.

Please enter your full name.

>>> *[USER INPUT]*

This will prompt the user to enter information during the account creation process. The text will change depending on whether the user is being prompted for their name, age, year group or password.

That is not a valid option. Please try again.

>>> *[USER INPUT]*

This will be presented to the user if they enter an invalid option. I have decided not to reprint the initial request so as not to clutter the screen.

Your username is: *[USERNAME]*

This will tell the user their newly created username.

# Log In

Username: *[USER INPUT]*

This will prompt the user to enter their username. I have decided for the user to enter their username alongside the ‘Username:’ text so give the program a more professional look.

Username: *[USERNAME ENTERED]*

Password: *[USER INPUT]*

This will prompt the user to enter their password. I have decided for the user to enter their username alongside the ‘Password:’ text so give the program a more professional look.

That user does not exist. Please try again.

>>> *[USER INPUT]*

This will be presented to the user if they enter an invalid username. The program will then request the username and password again, using the same interface described earlier.

That is not the correct password. Please try again.

>>> *[USER INPUT]*

This will be presented to the user if they enter an invalid passowrd. The program will then request the username and password again, using the same interface described earlier.

# User Report

Please enter the student’s username.

>>> *[USER INPUT]*

This will prompt the user to enter the username of the student’s account whose data they want to access.

That user does not exist. Please try again.

>>> *[USER INPUT]*

This will be presented to the user if they enter a username that is not stored in the ‘user\_info.json’ file. I have decided not to reprint the initial request so as not to clutter the screen.

User: Gemma Smith

Age: 17

Year: 12

Music: Grade 7

History: Grade 8

Computer Science: Grade 9

This is an example user report for the student given in the scenario. The grades are all in the same place on the screen for easy reading.

# Topic Report

This section will use the menu interface for the user to select which topic they want to access.

Computer Science

Average score: 4

Highest score: 5

Highest score achieved by Gemma Smith, aged 17, in year 12.

This is an example topic report for computer science, with the highest score being achieved by the student given in the scenario.

There is no high score data. Please complete a quiz then try again.

This shows when here is no high score data to report.

# Question

What note length is worth half a beat in simple quadruple time?

1. Hemidemisemiquaver
2. Minim
3. Quaver
4. Crotchet

>>> *[USER INPUT]*

This is an example question for music on hard difficulty. The amount of questions will change depending on the difficulty selected earlier in the program. The answers are indexed with letters so the user cannot confuse it with a menu.

That answer was correct!

This will be presented to the user if the answer the question correctly.

That answer was wrong! The correct answer was:

Quaver

I have decided not to include the lettered index because it is not needed for the user to see the correct answer.

# Quiz Finish

Score: 5/5

Percentage: 100%

Grade: 9

This section displays the users score, percentage, and grade for the quiz. All the information is in the same place for easy reading.

# Algorithms

# Starting Menu

This algorithm is contained within a function so that the program is modular and so that this section can be temporarily disabled during testing when needed to speed up the process. ‘log\_in()’, ‘new()’, and ‘reports()’ are all functions within the program.

1. FUNCTION starting\_menu()
   1. verified = False
   2. PRINT(“ 1) Log In\n 2) Create New Account\n 3) Generate Reports\n”)
   3. WHILE verified == False
      1. user\_in = INPUT()
      2. IF user\_in == “1” THEN
         1. verified = True
         2. log\_in()
      3. ELSEIF user\_in == “2” THEN
         1. verified = True
         2. new()
      4. ELSEIF user\_in == “3” THEN
         1. verified = True
         2. reports()
      5. ELSE
         1. PRINT(“\nThat is not a valid option. Please try again.\n”)
      6. ENDIF
   4. ENDWHILE
2. ENDFUNCTION

# Open ‘user\_info.json’

This algorithm opens and decodes the ‘user\_info.json’ file, and is used in both the ‘Log In’ section and the ‘Create New Account’ section. This algorithm is contained within a function so that the program is modular and can be used throughout the program. ‘RETURN()’ will store the variable contained within the brackets in the variable that the function has been called within. ‘DECODE()’ will decode a json file to a format usable in the program.

1. FUNCTION open\_users()
   1. file = OPENREAD(“user\_info.json”)
   2. user\_info = DECODE(file)
   3. RETURN(user\_info)
2. ENDFUNCTION

# Log In

This algorithm is contained within a function so that the program is modular and so that this section can be temporarily disabled during testing when needed to speed up the process. The ‘TRY’ & ‘EXCEPT’ statement runs the code in the ‘TRY’ block and if it encounters an error, will run the code in the ‘EXCPET’ block rather than terminating the program. topic\_menu() is a function within the program.

1. FUNCTION log\_in()
   1. user\_info = open\_users()
   2. validated = False
   3. WHILE validated == False
      1. username\_input = INPUT(“Username: “)
      2. password\_input = INPUT(“Password: “)
      3. TRY
         1. user = user\_info[username\_input]
         2. user\_exist = True
      4. EXCEPT
         1. PRINT(“That user does not exist. Please try again.”)
         2. user\_exist = False
      5. ENDTRY
      6. IF user\_exist == True
         1. password = user[“pass”]
         2. IF password == password\_input THEN
            1. validated = True
            2. topic\_menu()
         3. ELSE
            1. PRINT(“That is not the correct password. Please try again.”)
         4. ENDIF
      7. ENDIF
   4. ENDWHILE

# Generate Report

This algorithm is contained within a function so that the program is modular and so that this section can be temporarily disabled during testing when needed to speed up the process. ‘user\_report()’, and ‘topic\_report()’ are all functions within the program.

1. FUNCTION reports()
   1. verified = False
   2. PRINT(“ 1) User Report\n 2) Topic report\n”)
   3. WHILE verified == False
      1. user\_in = INPUT()
      2. IF user\_in == “1” THEN
         1. verified = True
         2. user\_report()
      3. ELSEIF user\_in == “2” THEN
         1. verified = True
         2. topic\_report()
      4. ELSE
         1. PRINT(“\nThat is not a valid option. Please try again.\n”)
      5. ENDIF
   4. ENDWHILE
2. ENDFUNCTION

# Generate Grade

This algorithm takes the score achieved by the user, and the total questions asked to generate a grade. This algorithm is contained within a function so that the program is modular and can be used throughout the program. ‘RETURN()’ will store the variable contained within the brackets in the variable that the function has been called within.

1. FUNCTON gen\_grade(score, totalQs)
   1. percent = score / totalQs
   2. IF percent > 0.9 THEN
      1. RETURN(9)
   3. ELSE IF percent > 0.8 THEN
      1. RETURN(8)
   4. ELSE IF percent > 0.7 THEN
      1. RETURN(7)
   5. ELSE IF percent > 0.6 THEN
      1. RETURN(6)
   6. ELSE IF percent > 0.5 THEN
      1. RETURN(5)
   7. ELSE IF percent > 0.4 THEN
      1. RETURN(4)
   8. ELSE IF percent > 0.3 THEN
      1. RETURN(3)
   9. ELSE IF percent > 0.2 THEN
      1. RETURN(2)
   10. ELSE IF percent > 0.1 THEN
       1. RETURN(1)
   11. ELSE
       1. RETURN(0)
   12. ENDIF

# User Report

This algorithm is contained within a function so that the program is modular and so that this section can be temporarily disabled during testing when needed to speed up the process. The ‘TRY’ & ‘EXCEPT’ statement runs the code in the ‘TRY’ block and if it encounters an error, will run the code in the ‘EXCPET’ block rather than terminating the program. The spaces being added to the string ‘line’ starting at 1.c.v.5. are to space the grade correctly. ‘gen\_grade()’ is a function within the program.

1. FUNCTION user\_report()
   1. user\_info = open\_users()
   2. validated = False
   3. WHILE validated == False
      1. username\_input = INPUT(“Please enter the student’s username.\n“)
      2. TRY
         1. user = user\_info[username\_input]
         2. validated = True
      3. EXCEPT
         1. PRINT(“That user does not exist. Please try again.”)
         2. validated = False
      4. ENDTRY
      5. IF validated == True THEN
         1. PRINT(“User: “,user[“name”])
         2. PRINT(“Age: “,user[“age”])
         3. PRINT(“Year: “,user[“year”],”\n”)
         4. quizzes = user[“quizzes”]
         5. FOR index = 0 to quizzes.LENGTH
            1. quiz = quizzes[index]
            2. line = quizzes[“topic”] + “:”
            3. IF quizzes[“topic”] == “Music” THEN

line = line + “ “

* + - * 1. ELSE IF quizzes[“topic]” == History” THEN

line = line + “ “

* + - * 1. ELSE

line = line + “ “

* + - * 1. ENDIF
        2. line = line + “Grade “
        3. score = quizzes[“score”]
        4. totalQs = quizzes[“totalQs”]
        5. line = line + gen\_grade(score, totalQs)
        6. PRINT(line)
      1. NEXT
    1. ENDIF
  1. ENDWHILE

# Topic Report

This algorithm is contained within a function so that the program is modular and so that this section can be temporarily disabled during testing when needed to speed up the process. The ‘TRY’ & ‘EXCEPT’ statement runs the code in the ‘TRY’ block and if it encounters an error, will run the code in the ‘EXCPET’ block rather than terminating the program.

1. FUNCTION topic\_report()
   1. user\_info = open\_users()
   2. TRY
      1. user = user\_info[“top”]
      2. validated = True
   3. EXCEPT
      1. PRINT(“There is no high score data. Please complete a quiz then try again.”)
      2. validated = False
   4. ENDTRY
   5. IF validated == True THEN
   6. ENDIF