TASKS

A console-based quiz in python has been implemented herein. The questions fall under the geography category asked in a random manner, and only testing common knowledge on the capital cities of countries.

## Quiz Structure

The test question source data are stored within a python dictionary, with the country name as the key and the capital city as the value. The key, value pair is accessed to ask the user of the capital city, with the user answer compared to value and marked as either correct or wrong. The quiz can be launched via the command line **$ python -version -m module file**, with the step-by-step prompt easy to use.

Steps after launch for the user to follow:

1. The user has to input their name, with the prompt clearly stating **“What’s your name?**”. The name has to be a string, with no digits and should not be an empty string, for which otherwise it is marked as an invalid name, and the user has to enter a valid name to proceed.
2. The user then selects the number of desired questions, via the request **“How many questions?”**. The input has to be a valid integer, that will determine the number of questions asked, with selection being as much as the user wants.
3. The questions are supplied via the **“What is the capital city of {country name}?”,** in successive manner, each question asked after the user supplies the answer as a string. The selection of the country name is random, via the **choice()** method of module random, on the data dictionary keys, country name. The capital city answer is supplied via input of string, with options of the answer being either of all lower case, uppercase, sentence case or title case, with the input converted to title case after stripping leading whitespace. The conversion to title case is due to the storage format of the capital city names in title case, with first letters of words in capital. The use of non-standard case input considers that the user can fail to use standard case which may render answer incorrect in certain instance.

Capital city names are assumed to contain no numbers, as such only alphabetical letters are required to make up the capital city name. An input of a number or empty space or any other symbol is regarded to be a request by the user to skipping question.

1. After answering all the questions, a list of the user data is stored, each trial using the following keys, within a dictionary detailing:
   1. 'Name': the name of the user
   2. 'No.': the question serial number unique to user
   3. 'Correct': string to outline whether the answer is Correct or Wrong depending on the answer provided.
   4. 'Question': states the country name, for which answer on the capital city is required.
   5. 'Answer': the user’s input on the capital city
   6. 'Correct Answer': a printout of the correct city name as expected by the question.
   7. 'Total Score': defines the score of the unique user, in comparison to the number of questions they are required to answer, as a string in format of scores/number of questions that user wished to answered
   8. 'Perc. Score': defines the score as a percentage of the number of questions answered, and derived as **100\*scores/question number**, with the value depending on the scores and questions answered.
   9. 'Mean Score': defines the average score of already answered questions at a given question number, derived as **scores/question number**

The score value is one (1) per correct question and zero (0) for a wrong answer supplied, as such the user ought to ensure they supply correct answers.

1. On completion,
   1. the summary of the player user data, and a subset version of the player statistics on the questions they got correctly or incorrectly (as well as showing the correct answer for any questions they answered incorrectly is provided.
   2. a prompt comes up to enquire whether a re-play is required through **“Would you like to play again? (Yes/No): “,** with desired answers of **“['Y', 'y', 'Yes', 'yes']:”** re-starting the whole process for a new user. Any other response is a confirmation of desire to exit the play.
   3. In the case a new user decides to start play, they go through the same steps as stated above, with the completion for all users leading to the printing of
      1. questions that all users got correct and which they got incorrect (as well as showing the correct answer for any questions they answered incorrectly
      2. questions they got correct and which they got incorrect (as well as showing the correct answer for any questions they answered incorrectly
      3. The name of the user with the highest score (as well as other users’ score) and the average score of all users.

The quiz is guided by the randomness of country names, as such the probability of a single user being asked the same question consecutively or in a single sitting is very low, giving the quiz program an area of its strength. In terms of weakness, the quiz questions are not limited, as such the user can select as much questions as they desire, with time to finish being the limiting factor.

## Code Structure

The code takes advantage of the object oriented programming by defining a **Quiz class** with methods:

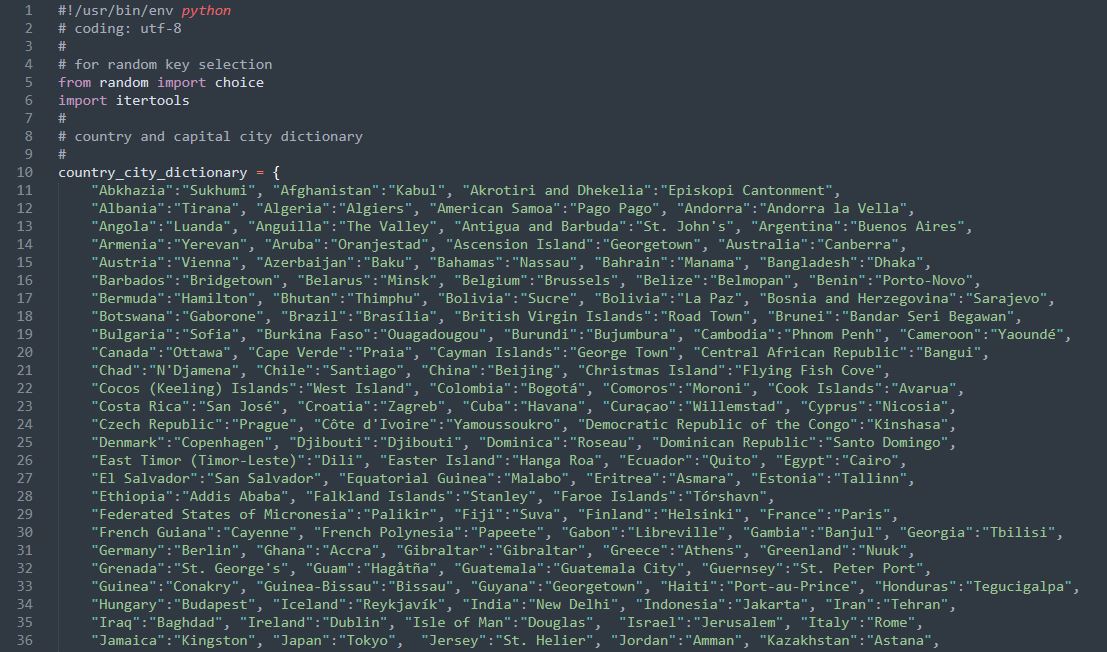
1. The user name via pla\_yer\_name()
2. The city input via country\_city\_input()
3. The individual player score printing via print\_player\_score()
4. All players score printing via print\_all\_player\_score()
5. The desired number of questions via num\_of\_questions()
6. The quiz() method puts together all methods for smooth quiz running.

Python data structures and algorithms were used, with the use of standard data structures strings, dictionaries to store and access data, string formatting to print out the data well, the use of try-Except to capture and validate errors arising during input of desired user variable, while loops and if-else conditions for question and input controls and class and associated methods structure.

## Appendix

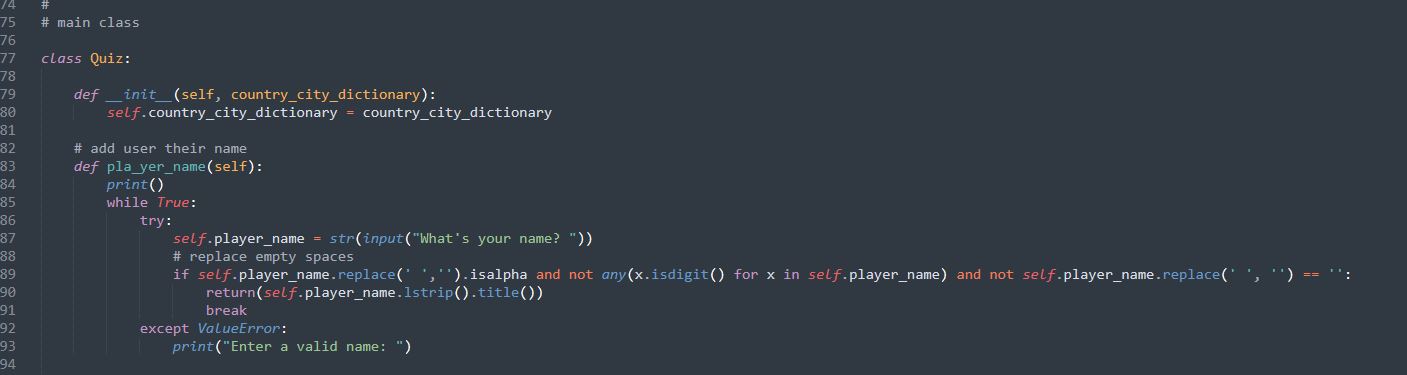
### Data Structure

The key, value pair of country, city data is provided at the start of code snippet. The test question source data are stored within a python dictionary, with the country name as the key and the capital city as the value. The key, value pair is accessed to ask the user of the capital city, with the user answer compared to value and marked as either correct or wrong.

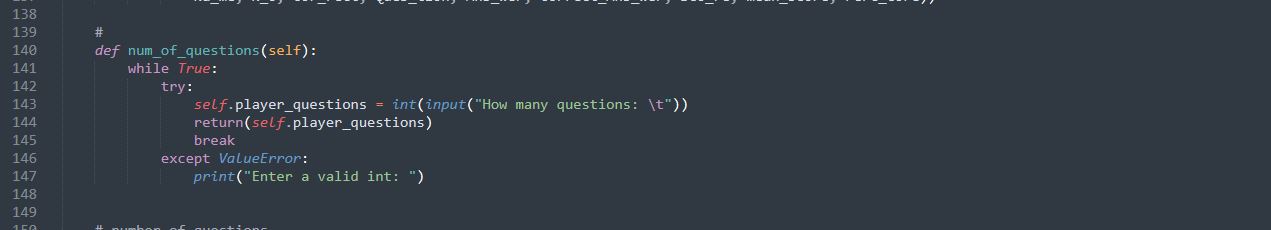


### Input Name and Number of Questions: 1: name via pla\_yer\_name() and num\_of\_questions()

The user has to input their name, with the prompt clearly stating **“What’s your name?**”. The name has to be a string, with no digits and should not be an empty string, for which otherwise it is marked as an invalid name, and the user has to enter a valid name to proceed.



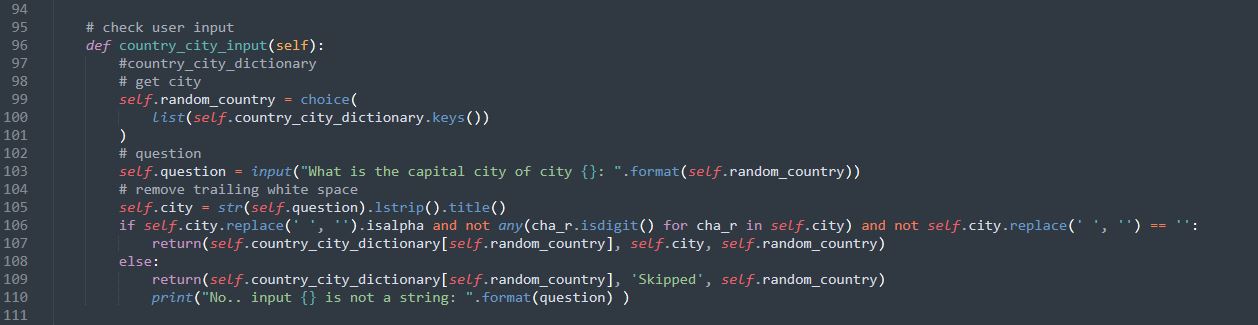
The user then selects the number of desired questions, via the request **“How many questions?”**. The input has to be a valid integer, that will determine the number of questions asked, with selection being as much as the user wants.



### Country Input: country\_city\_input()

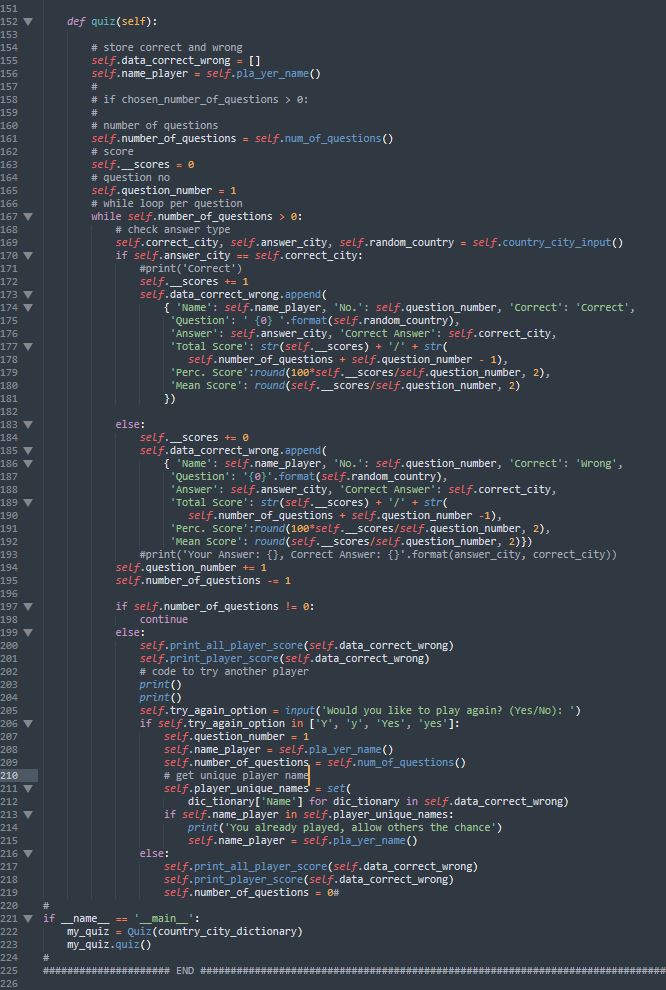
The questions are supplied via the **“What is the capital city of {country name}?”,** in successive manner, each question asked after the user supplies the answer as a string. The selection of the country name is random, via the **choice()** method of module random, on the data dictionary keys, country name. The capital city answer is supplied via input of string, with options of the answer being either of all lower case, uppercase, sentence case or title case, with the input converted to title case after stripping leading whitespace. The conversion to title case is due to the storage format of the capital city names in title case, with first letters of words in capital. The use of non-standard case input considers that the user can fail to use standard case which may render answer incorrect in certain instance.

Capital city names are assumed to contain no numbers, as such only alphabetical letters are required to make up the capital city name. An input of a number or empty space or any other symbol is regarded to be a request by the user to skipping question.



### User Data

The user data is supplied via quiz() method, where all user data is captured and stored.

****

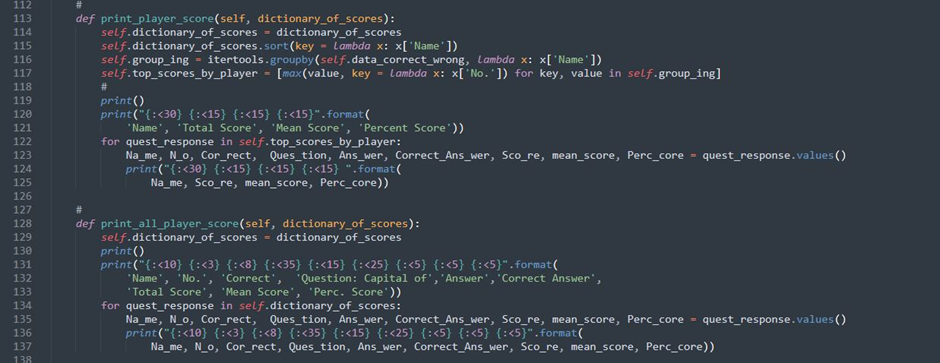
After answering all the questions, a list of the user data is stored, each trial using the following keys, within a dictionary detailing:

1. 'Name': the name of the user
2. 'No.': the question serial number unique to user
3. 'Correct': string to outline whether the answer is Correct or Wrong depending on the answer provided.
4. 'Question': states the country name, for which answer on the capital city is required.
5. 'Answer': the user’s input on the capital city
6. 'Correct Answer': a printout of the correct city name as expected by the question.
7. 'Total Score': defines the score of the unique user, in comparison to the number of questions they are required to answer, as a string in format of scores/number of questions that user wished to answered
8. 'Perc. Score': defines the score as a percentage of the number of questions answered, and derived as **100\*scores/question number**, with the value depending on the scores and questions answered.
9. 'Mean Score': defines the average score of already answered questions at a given question number, derived as **scores/question number**

The score value is one (1) per correct question and zero (0) for a wrong answer supplied, as such the user ought to ensure they supply correct answers.

### On completion of running quiz(): print\_player\_score() and print\_all\_player\_score()

1. the summary of the player user data, and a subset version of the player statistics on the questions they got correctly or incorrectly (as well as showing the correct answer for any questions they answered incorrectly is provided.
2. a prompt comes up to enquire whether a re-play is required through **“Would you like to play again? (Yes/No): “,** with desired answers of **“['Y', 'y', 'Yes', 'yes']:”** re-starting the whole process for a new user. Any other response is a confirmation of desire to exit the play.
3. In the case a new user decides to start play, they go through the same steps as stated above, with the completion for all users leading to the printing of
   1. questions that all users got correct and which they got incorrect (as well as showing the correct answer for any questions they answered incorrectly
   2. questions they got correct and which they got incorrect (as well as showing the correct answer for any questions they answered incorrectly
   3. The name of the user with the highest score (as well as other users’ score) and the average score of all users.



## System Demonstration

The quiz can be launched via the command line **$ python -version -m module file**, with the step-by-step prompt easy to use.

Steps after launch for the user to follow:

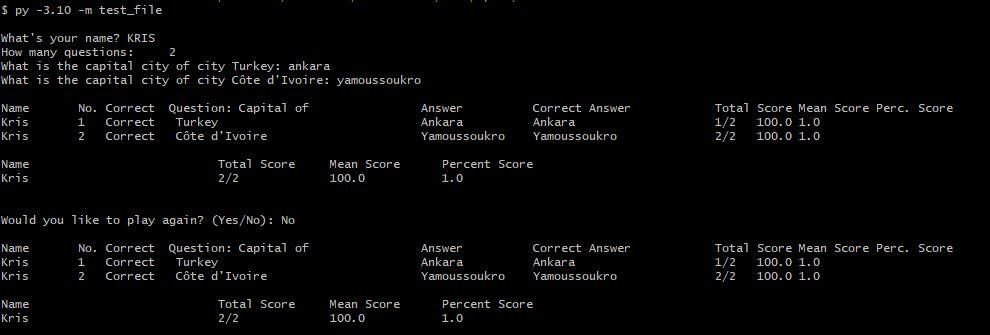


Figure 1 Only 1 user

The user has to input their name, number of desired questions. The questions are supplied via the in successive manner, with each question asked after the user supplies the answer as a string. After answering all the questions, a list of the user data is stored, the name of the user, the question serial number unique to user, string to outline whether the answer is Correct or Wrong depending on the answer provided, he country name, for which answer on the capital city is required, user’s input and the correct city name as expected, the score of the unique user, in comparison to the number of questions they are required to answer, the score as a percentage of the number of questions answered, and the average score of already answered questions at a given question number. On completion**,** the summary of the player user data, and a subset version of the player statistics on the questions they got correctly or incorrectly (as well as showing the correct answer for any questions they answered incorrectly is provided. A new user can be prompted to start the game with the final output being the name of the user with the highest score (as well as other users’ score) and the average score of all users as shown in the Figure 2 Multi user. The handling of exceptions, invalid input can be seen from the output in Figure 3 Quiz with error and invalid response handling,

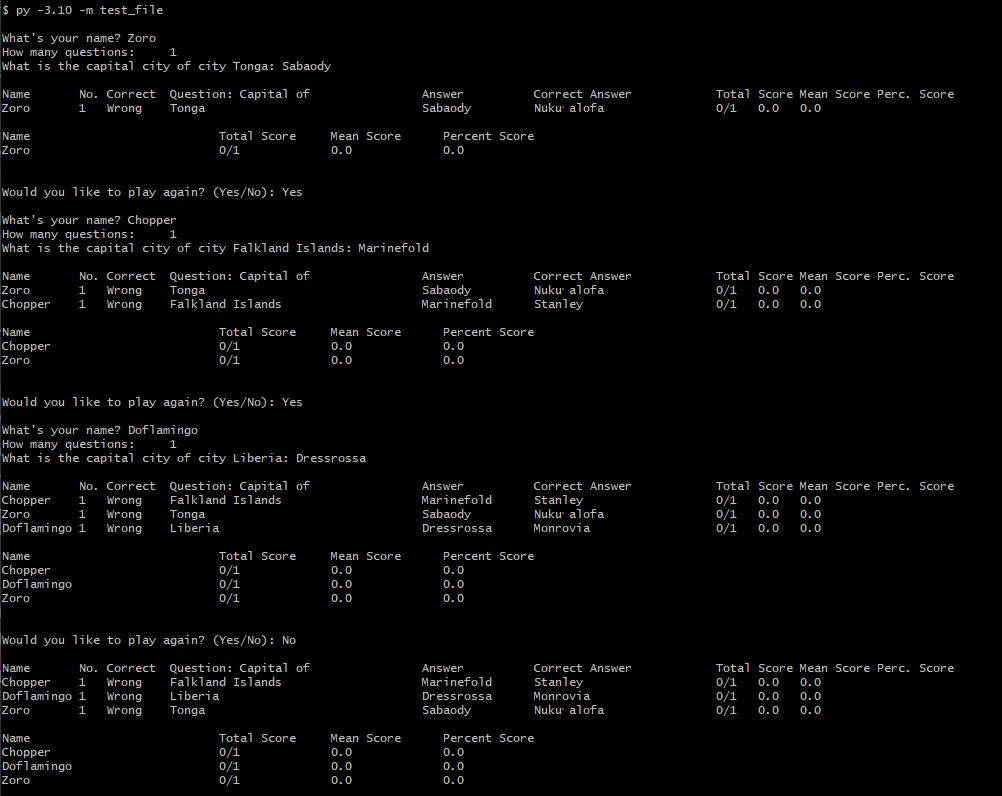


Figure 2 Multi user

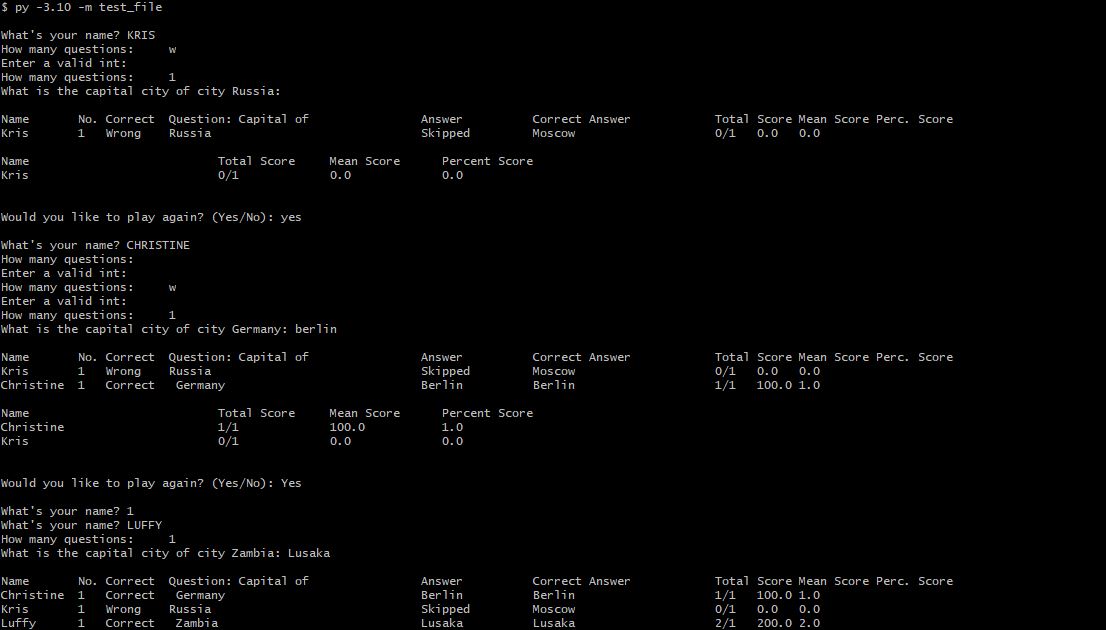


Figure 3 Quiz with error and invalid response handling