Quiz Game Project

Group 1

Group Members

Stavrinopoulos Viktor

Skamagkoulis Nikolaos

Program Overview

The program is a quiz game based on 3 difficulty levels and multiple question categories (General Knowledge, Geography, History, Music, Movies & TV Series, Science, Sports). Each game consists of 5 multiple choice (4 possible answers) questions, based on the chosen difficulty and category by the player. At the end of the game the program shows to the player the statistics of his performance, awards points according to his performance and provides a Text-Box, so the player can give his username and save his score in the Ranking Table. If the player is new to the quiz game, then a new entry is saved in the Table, otherwise his new score is added to his previous scores and climbs higher in the Ranking Table.

Program Structure & Functionality

Tkinter was used for the GUI of the program. The program consists of 5 main classes:

- **Start_Menu** class, which, when it is called, it clears the root window and creates two buttons (Start Quiz and Ranking).
- **QuizGame** class, which initializes the quiz when it is called.
- **Easy** class
- **Medium** class
- **Hard** class

The classes Easy, Medium and Hard are used to store the different categories in subclasses and each subclass contains a nested list of questions. There are 7 categories in total and each category has a sample of 10 questions stored in a nested list, so in total there are 210 questions available (70 for each difficulty).

When running the code, a new "Quiz Game" window appears showing 2 buttons, Start Quiz and Ranking (See Figure 1).

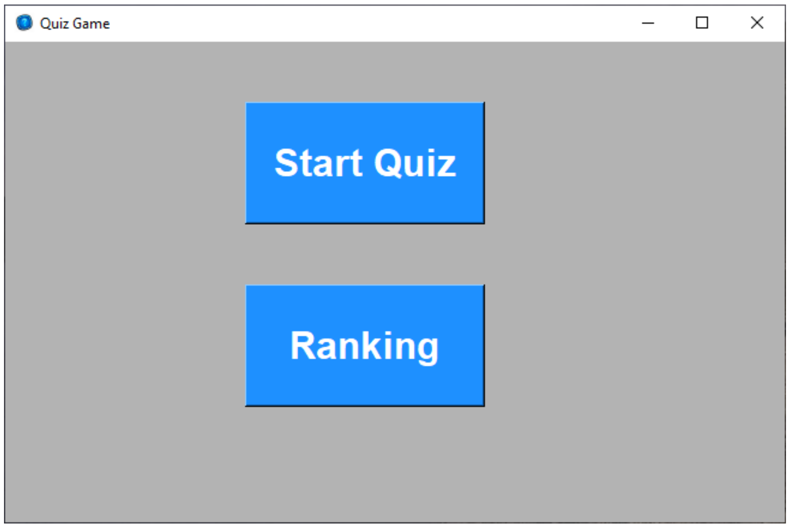


Figure 1: The main menu window with 2 buttons.

For the “Start Quiz” button a function “choose_difficulty()” has been created, so when the “Start Quiz” button is clicked it will call this function. Then “choose_difficulty()” function calls another function (clear_root()) to destroy all the widgets in the window and creates one Label (which contains text) and 3 buttons (one for each difficulty) (See Figure 2).

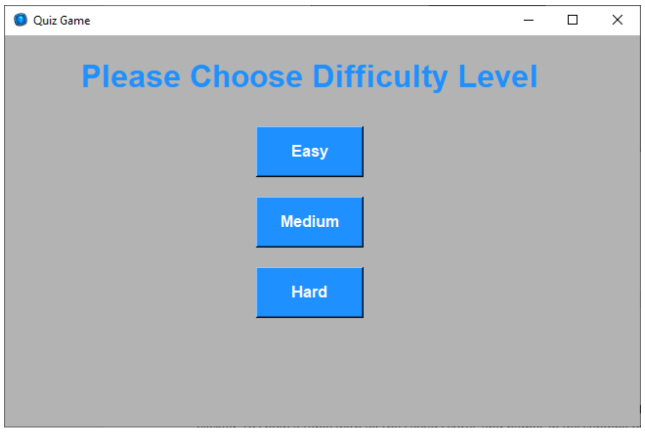


Figure 2: Window to choose difficulty level of the questions asked in the game.

Then, each of these 3 buttons has their own functionality (e.g. when clicking on Easy button only the questions that belong to this difficulty would be available and not questions from Medium or Hard).

So, in order to do so 3 classes were created, one for each difficulty. Each class has 7 subclasses, one for each category and inside those 7 subclasses the questions are stored in a nested list. Every question is stored inside a sublist. In index 0 of this sublist is the string of the question, while in indexes 1,2,3,4 of this list are the strings of the answers. The correct answer is always at index 1.

Then, 3 functions were created (one for each button, choose_easy_category(), choose_medium_category() and choose_hard_category()). When one of these 3 functions is called, it destroys all widgets from the window and creates a Label (which contains text) and 7 buttons (one for each category to choose) (See Figure 3).

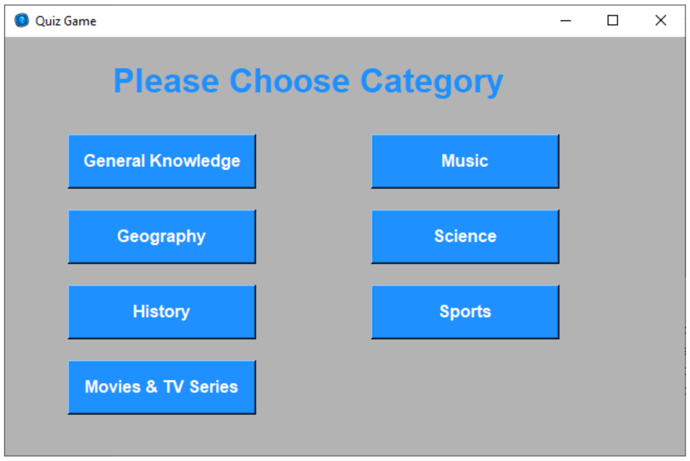
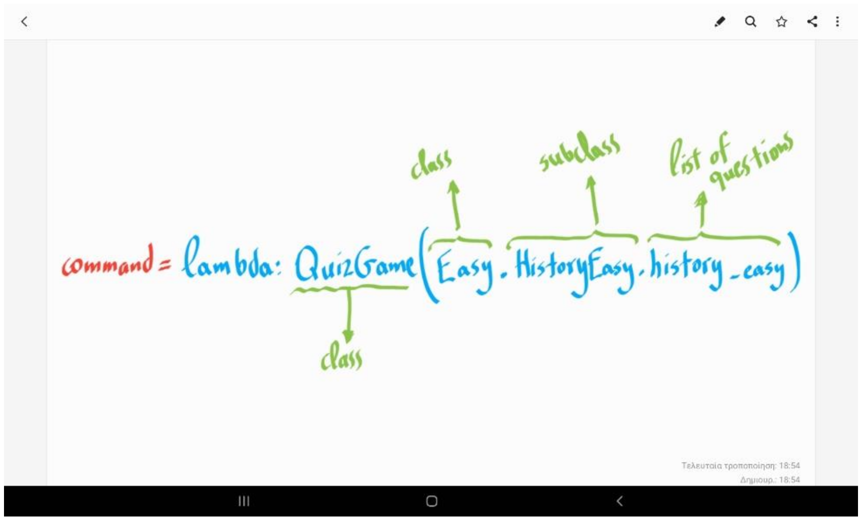


Figure 3: Window to choose the category of the questions asked in the game.

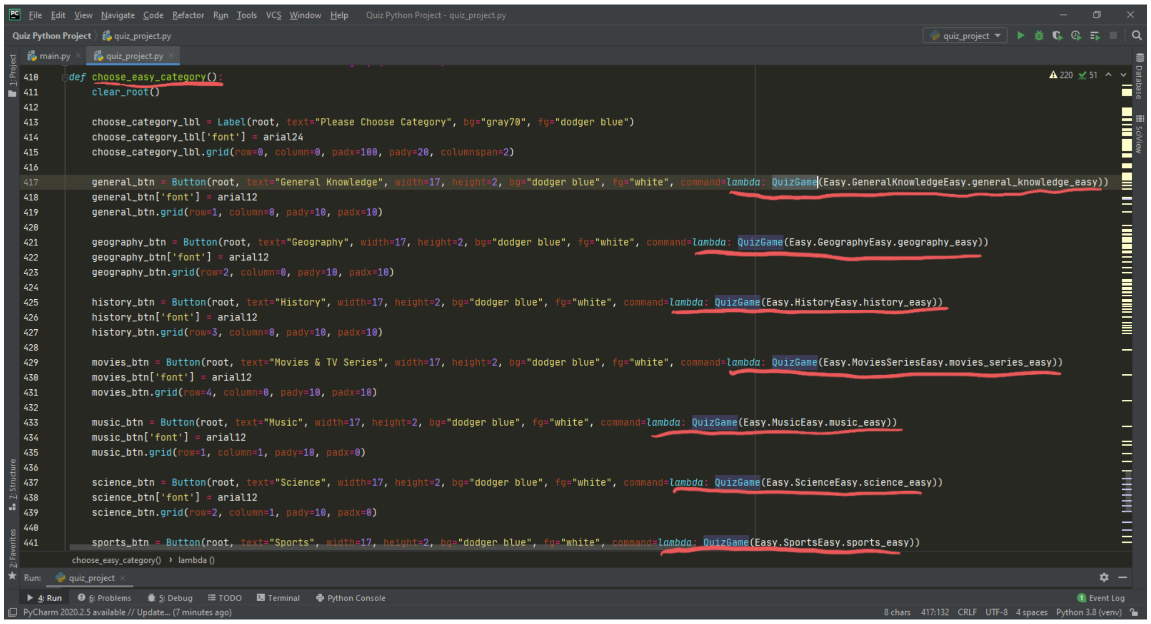
Just like in the previous window with the difficulties these 7 buttons to have their own functionality when clicked (e.g. when clicking on History button only the questions that belong to this category would be available and not questions from another and also only questions from the difficulty that was chosen in the previous step). So, when one of these buttons is clicked the Quiz starts, by calling the constructor of the QuizGame class, which takes as parameters a list. This list is going to be a list of questions and it will correspond to the choices that the player made regarding the difficulty and category. In order to pass a parameter to the QuizGame constructor through the command of the button, a lambda function has to be used. As parameter is given the correct list by accessing the class and subclass (e.g. if the player chose Easy and History then the command of the button equals this: See Figure 4).



```
command = lambda: QuizGame(Easy.HistoryEasy.history_easy)
```

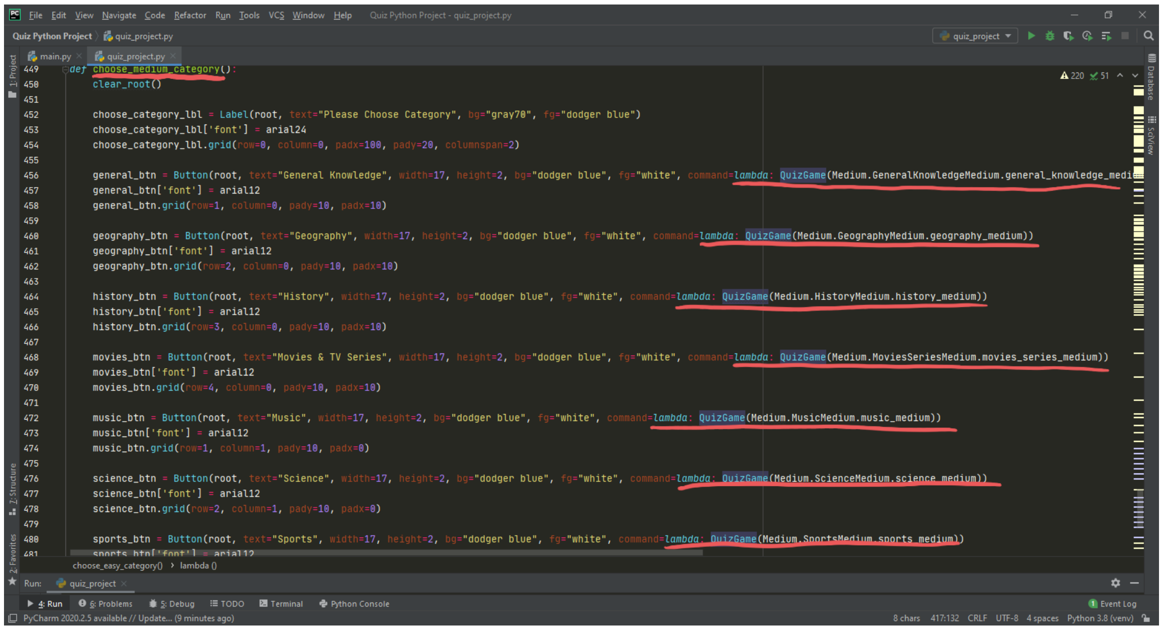
Figure 4: Example of command of the button history_btn.

The same thing goes for every button and for all the tree functions (choose_easy_category(), choose_medium_category() and choose_hard_category()) (See Figures 5 to 7).



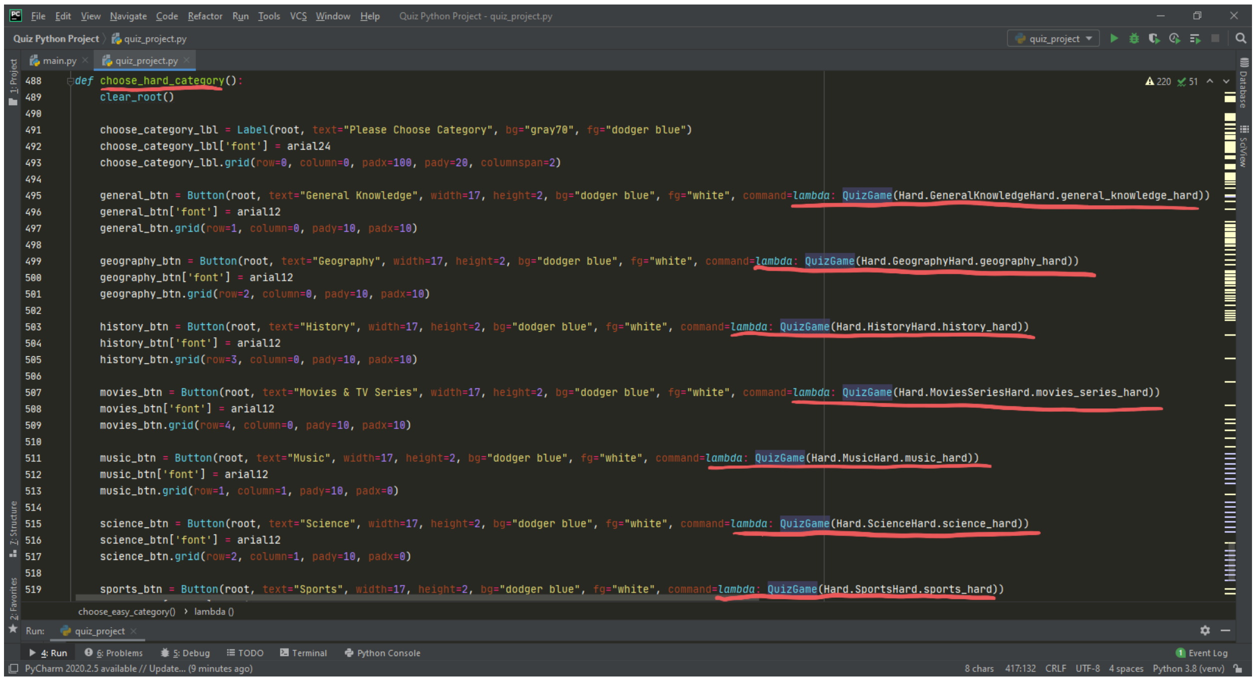
```python
def choose_easy_category():
    clear_root()

    choose_category_lbl = Label(root, text="Please Choose Category", bg="gray70", fg="dodger blue")
    choose_category_lbl['font'] = arial24
    choose_category_lbl.grid(row=0, column=0, padx=100, pady=20, columnspan=2)

    general_btn = Button(root, text="General Knowledge", width=17, height=2, bg="dodger blue", fg="white", command=lambda: QuizGame(Easy.GeneralKnowledgeEasy.general_knowledge_easy))
    general_btn['font'] = arial12
    general_btn.grid(row=1, column=0, pady=10, padx=10)

    geography_btn = Button(root, text="Geography", width=17, height=2, bg="dodger blue", fg="white", command=lambda: QuizGame(Easy.GeographyEasy.geography_easy))
    geography_btn['font'] = arial12
    geography_btn.grid(row=2, column=0, pady=10, padx=10)

    history_btn = Button(root, text="History", width=17, height=2, bg="dodger blue", fg="white", command=lambda: QuizGame(Easy.HistoryEasy.history_easy))
    history_btn['font'] = arial12
    history_btn.grid(row=3, column=0, pady=10, padx=10)

    movies_btn = Button(root, text="Movies & TV Series", width=17, height=2, bg="dodger blue", fg="white", command=lambda: QuizGame(Easy.MoviesSeriesEasy.movies_series_easy))
    movies_btn['font'] = arial12
    movies_btn.grid(row=4, column=0, pady=10, padx=10)

    music_btn = Button(root, text="Music", width=17, height=2, bg="dodger blue", fg="white", command=lambda: QuizGame(Easy.MusicEasy.music_easy))
    music_btn['font'] = arial12
    music_btn.grid(row=1, column=1, pady=10, padx=0)

    science_btn = Button(root, text="Science", width=17, height=2, bg="dodger blue", fg="white", command=lambda: QuizGame(Easy.ScienceEasy.science_easy))
    science_btn['font'] = arial12
    science_btn.grid(row=2, column=1, pady=10, padx=0)

    sports_btn = Button(root, text="Sports", width=17, height=2, bg="dodger blue", fg="white", command=lambda: QuizGame(Easy.SportsEasy.sports_easy))
```

Figure 5: Choose_easy_category() function.



```python
def choose_medium_category():
    clear_root()

    choose_category_lbl = Label(root, text="Please Choose Category", bg="gray70", fg="dodger blue")
    choose_category_lbl['font'] = arial24
    choose_category_lbl.grid(row=0, column=0, padx=100, pady=20, columnspan=2)

    general_btn = Button(root, text="General Knowledge", width=17, height=2, bg="dodger blue", fg="white", command=lambda: QuizGame(Medium.GeneralKnowledgeMedium.general_knowledge_medi
    general_btn['font'] = arial12
    general_btn.grid(row=1, column=0, pady=10, padx=10)

    geography_btn = Button(root, text="Geography", width=17, height=2, bg="dodger blue", fg="white", command=lambda: QuizGame(Medium.GeographyMedium.geography_medium))
    geography_btn['font'] = arial12
    geography_btn.grid(row=2, column=0, pady=10, padx=10)

    history_btn = Button(root, text="History", width=17, height=2, bg="dodger blue", fg="white", command=lambda: QuizGame(Medium.HistoryMedium.history_medium))
    history_btn['font'] = arial12
    history_btn.grid(row=3, column=0, pady=10, padx=10)

    movies_btn = Button(root, text="Movies & TV Series", width=17, height=2, bg="dodger blue", fg="white", command=lambda: QuizGame(Medium.MoviesSeriesMedium.movies_series_medium))
    movies_btn['font'] = arial12
    movies_btn.grid(row=4, column=0, pady=10, padx=10)

    music_btn = Button(root, text="Music", width=17, height=2, bg="dodger blue", fg="white", command=lambda: QuizGame(Medium.MusicMedium.music_medium))
    music_btn['font'] = arial12
    music_btn.grid(row=1, column=1, pady=10, padx=0)

    science_btn = Button(root, text="Science", width=17, height=2, bg="dodger blue", fg="white", command=lambda: QuizGame(Medium.ScienceMedium.science_medium))
    science_btn['font'] = arial12
    science_btn.grid(row=2, column=1, pady=10, padx=0)

    sports_btn = Button(root, text="Sports", width=17, height=2, bg="dodger blue", fg="white", command=lambda: QuizGame(Medium.SportsMedium.sports_medium))
    sports_btn['font'] = arial12
```

Figure 6: Choose_medium_category() function.



```python
def choose_hard_category():
    clear_root()

    choose_category_lbl = Label(root, text="Please Choose Category", bg="gray70", fg="dodger blue")
    choose_category_lbl['font'] = arial24
    choose_category_lbl.grid(row=0, column=0, padx=100, pady=20, columnspan=2)

    general_btn = Button(root, text="General Knowledge", width=17, height=2, bg="dodger blue", fg="white", command=lambda: QuizGame(Hard.GeneralKnowledgeHard.general_knowledge_hard))
    general_btn['font'] = arial12
    general_btn.grid(row=1, column=0, pady=10, padx=10)

    geography_btn = Button(root, text="Geography", width=17, height=2, bg="dodger blue", fg="white", command=lambda: QuizGame(Hard.GeographyHard.geography_hard))
    geography_btn['font'] = arial12
    geography_btn.grid(row=2, column=0, pady=10, padx=10)

    history_btn = Button(root, text="History", width=17, height=2, bg="dodger blue", fg="white", command=lambda: QuizGame(Hard.HistoryHard.history_hard))
    history_btn['font'] = arial12
    history_btn.grid(row=3, column=0, pady=10, padx=10)

    movies_btn = Button(root, text="Movies & TV Series", width=17, height=2, bg="dodger blue", fg="white", command=lambda: QuizGame(Hard.MoviesSeriesHard.movies_series_hard))
    movies_btn['font'] = arial12
    movies_btn.grid(row=4, column=0, pady=10, padx=10)

    music_btn = Button(root, text="Music", width=17, height=2, bg="dodger blue", fg="white", command=lambda: QuizGame(Hard.MusicHard.music_hard))
    music_btn['font'] = arial12
    music_btn.grid(row=1, column=1, pady=10, padx=0)

    science_btn = Button(root, text="Science", width=17, height=2, bg="dodger blue", fg="white", command=lambda: QuizGame(Hard.ScienceHard.science_hard))
    science_btn['font'] = arial12
    science_btn.grid(row=2, column=1, pady=10, padx=0)

    sports_btn = Button(root, text="Sports", width=17, height=2, bg="dodger blue", fg="white", command=lambda: QuizGame(Hard.SportsHard.sports_hard))
```

File Edit View Navigate Code Refactor Run Tools VCS Window Help
Quiz Python Project - quiz_project.py
Quiz Python Project
quiz_project.py
main.py
quiz_project.py
choose_easy_category() › lambda ()
Run: quiz_project
4: Run
6: Problems
5: Debug
TODO
Terminal
Python Console
Event Log
PyCharm 2020.2.5 available // Update... (9 minutes ago)
8 chars 417:132 CRLF UTF-8 4 spaces Python 3.8 (venv)

Figure 7: Choose_hard_category() function.

The QuizGame class constructor when called, with a list of questions as parameter, initializes the quiz. Once again the clear_root() function destroys the widgets to clear the window and then define some variables and buttons (without placing them inside the root yet) that will be used later, and lastly call a function Question() to start generating questions for the game.

The Question() function generates random questions from the given list(from the parameter), using the randint() function of the random library, and shuffles their answers (answers are found in indexes 1 to 4 in every sublist) and stores them in an answer list. It also creates 4 buttons (answer1_button, answer2_button, answer3_button, answer4_button) each for each answer and some functions that give functionalities to these buttons (button1_click_functionality(), button2_click_functionality(), button3_click_functionality(), button4_click_functionality()). For example, when an answer button is clicked if that was the right answer, then the color of the button turns green otherwise it turns red and the right answer turns green. Also, after an answer button has been clicked all answer button are disabled and the only one available to click is the “Next” to show the next question.

After 5 questions have been answered, the game is finished and it shows the results of the game, the achieved score, which is calculated by multiplying the correct answers by 10 and provides the option to give a name of the player and save his score to the ranking. If the player gives a name and click save, then his name and his score are written in a text file using the function save (called by the command of the save button). After the name and the score have been written in the text file, another function is called inside the save function in order to sort the contents of the file and remove duplicates, in case there some. If the player plays for the first time a new entry will be made with his username and score, otherwise the duplicate name will be removed but his new score will be added to his previous one.

The sorted contents are then written in another text file along with enumeration in descending order, which will be read and projected in a new window when the Ranking button from the Start Menu is clicked. The Ranking window also has a scrollbar, because the length of the text file may bee longer than the fixed dimensions of the quiz game window.

Work Division

Functionalities of the project had been decided from both of us at the beginning of this course, but the code was written solely by Stavrinopoulos Viktor. Nikolaos Skamagkoulis did not provide any code for this project.

Stavrinopoulos Viktor (100%)

Skamagkoulis Nikolaos (0%)

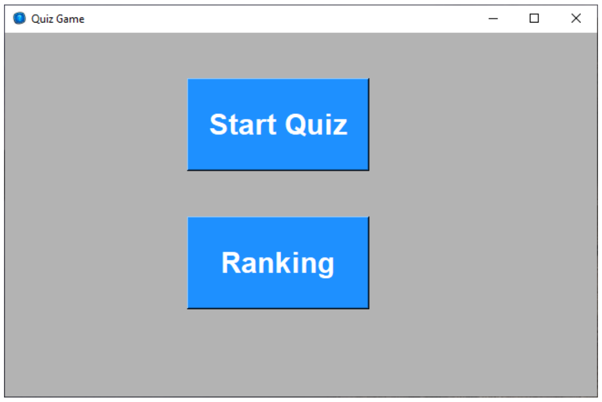


Figure 8: Main Menu

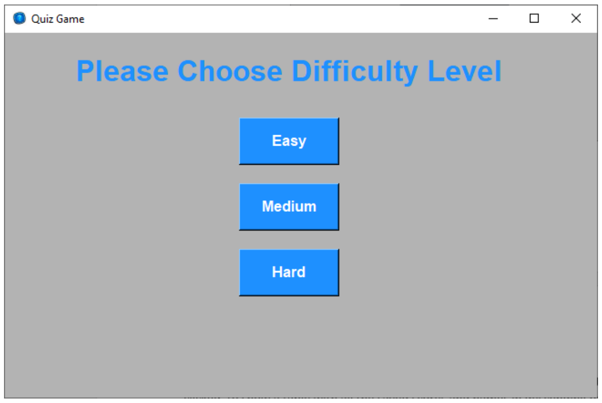


Figure 9: Choose difficulty window

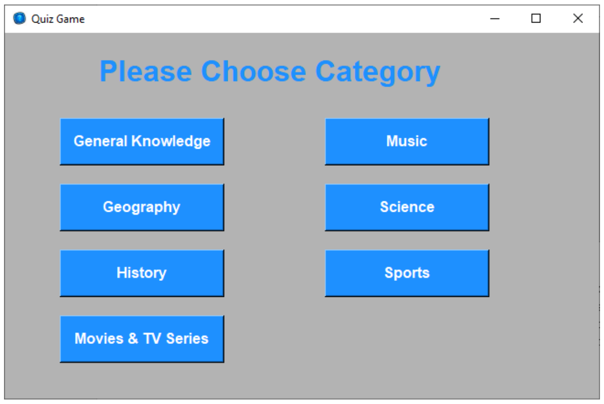


Figure 10: Choose category window

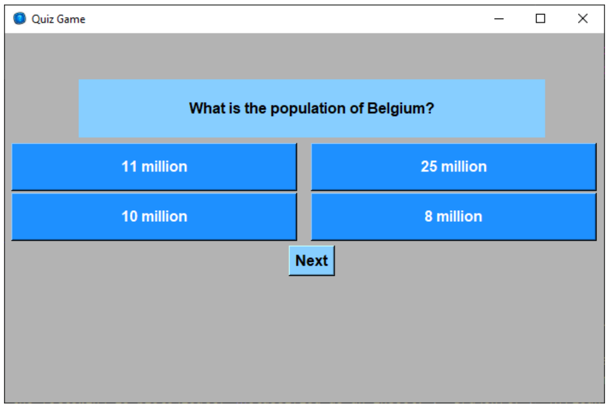


Figure 11: First question displayed

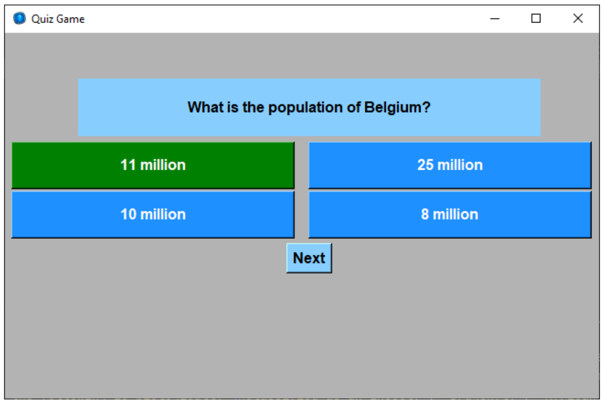


Figure 12: First question answered correctly

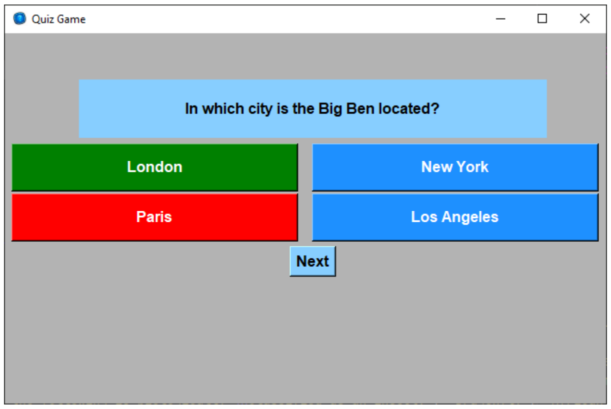
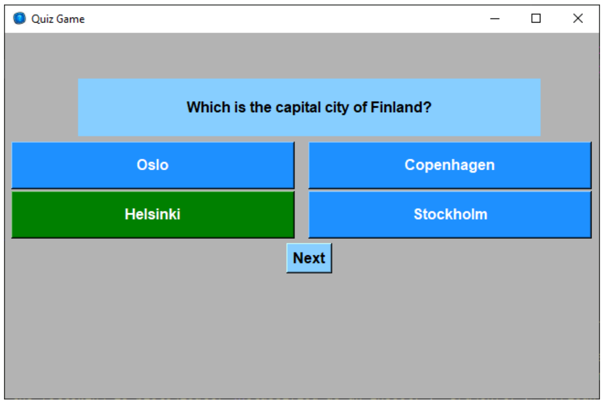
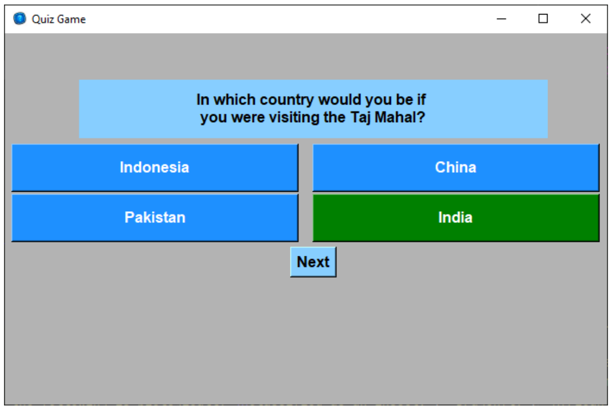


Figure 13: Second question answered wrong



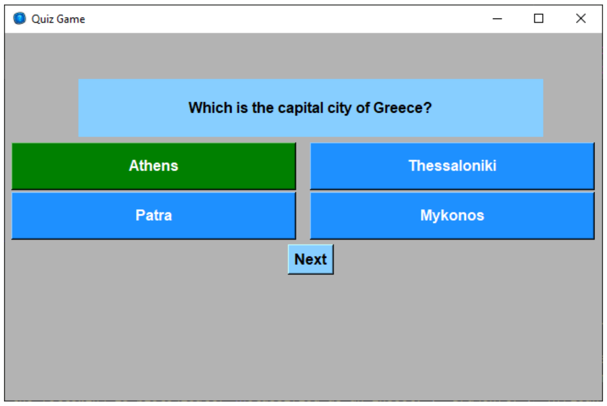
Which is the capital city of Finland?

Figure 14: Third question answered right



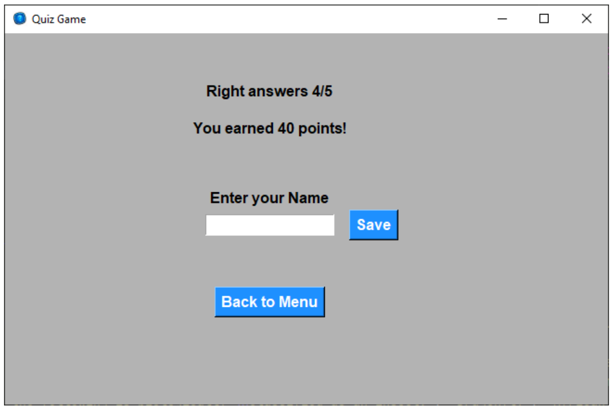
In which country would you be if you were visiting the Taj Mahal?

Figure 15: Fourth question answered right



Which is the capital city of Greece?

Figure 16: Fifth question answered right

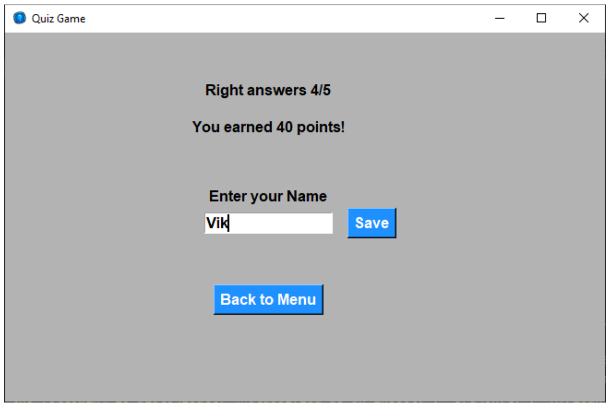


Right answers 4/5

You earned 40 points!

Enter your Name

Figure 17: Results window



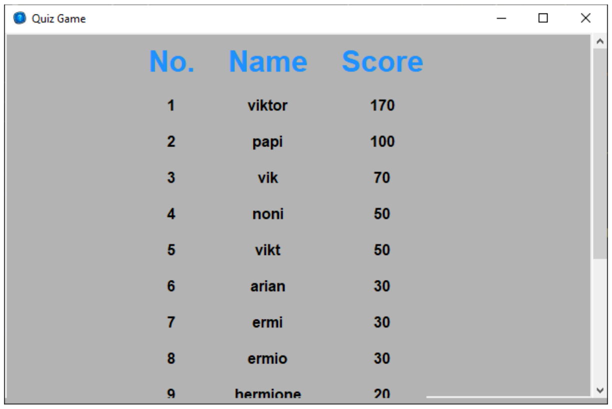
Right answers 4/5

You earned 40 points!

Enter your Name

Vik

Figure 18: Results window with given name



No.	Name	Score
1	viktor	170
2	papi	100
3	vik	70
4	noni	50
5	vikt	50
6	arian	30
7	ermi	30
8	ermio	30
9	hermione	20

Figure 19

A demonstration of a game of the Quiz Game.