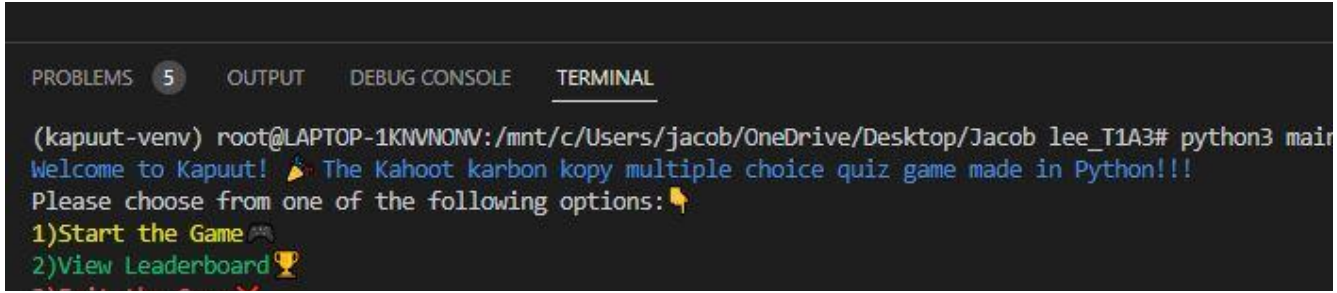
Introduction

Multiple Choice Quiz based on Kahoot - Kapuut Karbon Kopy

- User input for name
- A list of questions and multiple choice answers
- Answer choices are usually colored red, blue, green
- and yellow
- Score at end
- Leaderboard
- A way to start and end game
- timer

Main Page



main.py > ...

```python
from colorama import Fore, Style
from colored import fg, bg, attr
from rich.console import Console
import emoji

#
console = Console()

# Try to import the run_quiz and view_leaderboard functions from the quiz module
try:
    from quiz import run_quiz, view_leaderboard
# If the quiz module cannot be imported, print error message
except ImportError:
    print("Error: Could not import quiz module.")
# Define a function called print_menu that prints the game menu. Use Colorama package for color. Rich package for emojis.
def print_menu():
    print(Fore.BLUE + "Welcome to Kapuut! 🎉 The Kahoot karbon kopy multiple choice quiz game made in Python!!!" + Style.RESET_ALL)
    print("Please choose from one of the following options:👇")
    print(Fore.YELLOW + "1)" + Fore.YELLOW + "Start the Game🎮")
    print(Fore.GREEN + "2)" + Fore.GREEN +  "View Leaderboard🏆")
    print(Fore.RED + "3)" + Fore.RED +  "Exit the Game❌" + Style.RESET_ALL)
```

```python
def main_menu():
    # Loop indefinately
    while True:
        # Call the print_menu function to display menu
        print_menu()
        # Get user choice
        choice = input("Enter your choice (1-3): ")
        try:
            # If user chose option 1, start the game
            if choice == "1":
                # Call the start_game function and store its return value in play_again
                play_again = start_game()
                # If play_again False, thank user and break loop
                if not play_again:
                    print(Fore.GREEN + "Thank you for playing Kapuut! Hope to see you back here real soon" + Style.RESET_ALL)
                    print(emoji.emojize(":grinning_face_with_big_eyes:"))
                    break
                # If user chose option 2, view leaderboard
            elif choice == "2":
                view_leaderboard()
                # If user chose option 3, thank them and break out of loop
            elif choice == "3":
                print(Fore.YELLOW + "Thank you for playing hope to see you soon!!!" + Style.RESET_ALL)
                print(emoji.emojize(":grinning_face_with_big_eyes:"))
                break
            else:
                raise ValueError(
        except ValueError as e:
            print(f"error: {e}")
        except Exception as e:
            print(f"Error: {e}")
        exit()
```

```
(kapuut-venv) root@LAPTOP-1KNVNONV:/mnt/c/Users/jacob/OneDrive/Desktop/Jacob lee_T1A3# python3 main.py
Welcome to Kapuut! 🎉 The Kahoot karbon kopy multiple choice quiz game made in Python!!!
Please choose from one of the following options:👇
1)Start the Game🎮
2)View Leaderboard🏆
3)Exit the Game❌
Enter your choice (1-3): 1
Welcome to Kapuut Quiz Game!
You will be asked 7 multiple choice questions.
You will earn points for each correct answer.
Good luck ALL!!!

Enter your name: Jacob
```

Dictionary

dictionary.py

```
1   import string
2   #moved this from main.py to dictionary.py - storing the label and answer alternatives with zip() in a dictionary...
3
4   dict(zip(string.ascii_lowercase, ["He can start fires with his mind", "Super Speed", "He can turn invisible", "He can control lemmings by whistling"]))
5   {'a': 'He can start fires with his mind', 'b': 'Super Speed', 'c': 'He can turn invisible', 'd': 'He can control lemmings by whistling'}
6   dict(zip(string.ascii_lowercase, ["Seth MacFarlane", "Brian Peter Green", "Seth Rogan", "Bill Burr"]))
7   {'a': 'Seth MacFarlane', 'b': 'Brian Peter Green', 'c': 'Seth Rogan', 'd': 'Bill Burr'}
8   dict(zip(string.ascii_lowercase, ["Seven", "One", "Nine", "Five"]))
9   {'a': 'Seven', 'b': 'One', 'c': 'Nine', 'd': 'Five'}
10  dict(zip(string.ascii_lowercase, ["MacGyver", "Magnum P.I.", "Die Hard", "Harry Potter"]))
11  {'a': 'MacGyver', 'b': 'Magnum P.I.', 'c': 'Die Hard', 'd': 'Harry Potter'}
12  dict(zip(string.ascii_lowercase, ["Tae-Jitsu", "Tae-Bo", "Fight Club", "Tang-soo-do"]))
13  {'a': 'Tae-Jitsu', 'b': 'Tae-Bo', 'c': 'Fight Club', 'd': 'Tang-soo-do',}
14  dict(zip(string.ascii_lowercase, ["Mila Kunis", "Meg Ryan", "Jodie Foster", "Nancy Cartwright"]))
15  {'a': 'Mila Kunis', 'b': 'Meg Ryan', 'c': 'Jodie Foster', 'd': 'Nancy Cartwright'}
16  dict(zip(string.ascii_lowercase, ["Golf", "Badmington", "Frisbee Golf", "Synchronised Swimming"]))
17  {'a': 'Golf', 'b': 'Badmington', 'c': 'Frisbee Golf', 'd': 'Synchronised Swimming'}
```

Quiz Questions

questions.py > questions_dict

```
51      #General knowledge quiz questions and multiple choice answers
52      },
53      "What is age is the longest recorded age that an elephant has ever lived?": {
54          "alternatives": ["86 years", "17 years", "49 years", "142 years"],
55          "correct_answer": "86 years"
56
57      },
58      "What is a Tarsier?": {
59          "alternatives": ["A primate","A bird", "A marsupial", "A lizard"],
60          "correct_answer": "A primate"
61
62      },
63      "In darts, what's the most points you can score with a single throw?": {
64          "alternatives": ["60 points", "20 points", "50 points", "100 points"],
65          "correct_answer": "60 points"
66
67      },
```

Run Quiz Function

```
def run_quiz():
    try:
        # use input function to prompt the user to enter their name
        name = input("Enter your name: ")
        #set the initial value of high_score to zero
        high_score = 0
        # create an empty list called leaderboard
        leaderboard = []
        # use a while loop to keep running the quiz until the user chooses to stop
        while True:
            score = 0
            num_correct = 0
            # use the random sample function to select 7 random questions from the questions_dict dictionary
            random_questions = random.sample(list(questions_dict.items()), 7)
            # use a for loop to iterate over the enumerated random_questions list starting from index 1
            for num, (question, data) in enumerate(random_questions, start=1):
                question_score, answer_label, correct_answer, labeled_alternatives = play_question(num, question, data)
                score += question_score
                if check_answer(correct_answer, answer_label, labeled_alternatives):
                    num_correct += 1
             #
            high_score = update_leaderboard(score, name, high_score, leaderboard, num_correct, num)
            print_results(num_correct, score, len(random_questions), high_score)
            #
            if input("Do you want to play again? (y/n) ").lower() != 'y':
                #if their response is not y, break out of the while loop
                break
            # if a ValueError exception is raised during the execution of the try block, print out an error message
    except ValueError as e:
        print(f"Error: {e}")
    except Exception as e:
        print(f"Error: {e}")
        exit()
```

Leaderboard

```python
# Defines a function called load_leaderboard
def load_leaderboard():
    try:
        with open('leaderboard.txt', 'a+') as f:
            #Strip any whitespace from each line and Split each line by comma to seperate score and name
            leaderboard = [line.strip().split(',') for line in f.readlines()]
            #Convert each score to an interger and create a list of tuples with score and name
            leaderboard = [(int(score), name) for score, name in leaderboard]
            #Sort the list of tuples by score in descending order
            leaderboard.sort(reverse=True)
            from string import ascii_lowercase#Only keep the top 10 scores
            leaderboard = leaderboard[:10]
        return leaderboard
    except IOError:
        print("Error: Could not open leaderboard file")
```

leaderboard.txt

```
1 4865,Jacob
2
```

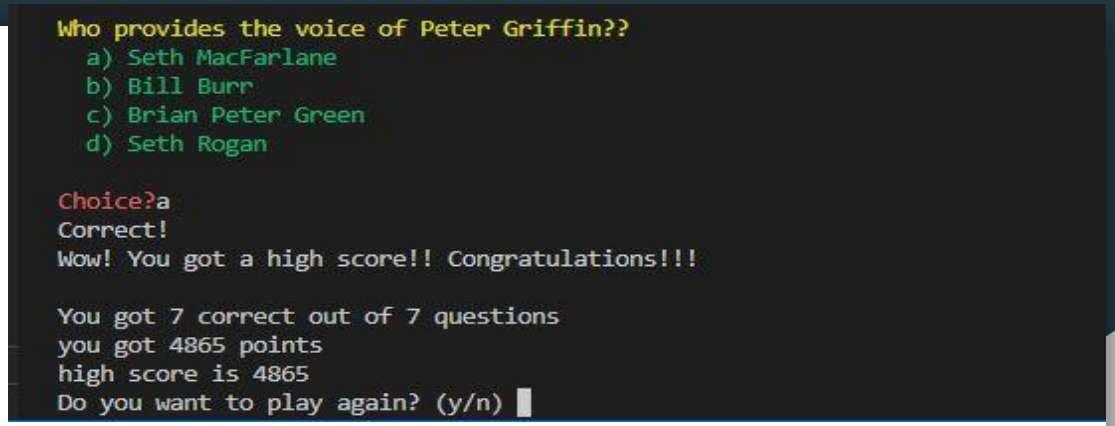
Update leaderboard

```python
def update_leaderboard(score, player_name, high_score, leaderboard, num_correct, num):

    # check if the leaderboard is empty or if tuple of (score, player_name) is greater than the last element in leaderboard
    if not leaderboard or score > leaderboard[-1][0]:
        #open a file called leaderboard.txt in write mode
        with open('leaderboard.txt', 'w') as f:
            # append the tuple to the leaderboard list
            leaderboard.append((score, player_name))
            #sort leaderboard list in descending order
            leaderboard.sort(reverse=True)
            # slice leaderboard list to include the top 10 scores or elements only
            leaderboard = leaderboard[:10]
            # create a new list of formatted strings for each element in leaderboard list
            leaderboard = [f"{score},{name}" for score, name in leaderboard]
            # writes the elements of the leaderboard list to the file seperated by newlines and add a newline at the end
            f.write('\n'.join(leaderboard) + '\n')

        if score > high_score:
            high_score = score
            print("Wow! You got a high score!! Congratulations!!!")
    return high_score
```

REVIEW



Issues - Scoring - The score states 7 out of 7 correct, even if you answer a question wrong, and gives you max score of 7x695 so everyone gets a high score essentially

Modulising the code and defining functions

Pytest

Timer - no go