l6-ifcs-formative-python-quiz README

This project is a modular Python quiz application with a graphical interface. It is designed for formative assessment in the Intensive Foundations of Computer Science and Programming course.

User Documentation

What is this?

This is a simple quiz app. When you run it, a window will appear with questions and a text input field for answers. Type your answer and click the "Submit" button. At the end, your score will be shown.

How do I use it?

1. Install dependencies if you do not already have them.
2. Run the quiz app
3. Answer the questions as they appear.
4. At the end, see your score!

Installing Dependencies

You need Python 3.x installed. Tkinter is usually included with Python. If you have Python, you likely have everything you need.

To install Python, visit: https://www.python.org/downloads/

How to Run the Quiz App

1. Download or clone this repository to your computer.
2. Open a terminal (Command Prompt on Windows, Terminal on macOS/Linux) and navigate to the project folder. Note if the `l6-ifcs-formative-python-quiz` folder is nested inside other folders, you will need to specify this in the path below before the repository folder and separate them by `/`:
   `cd l6-ifcs-formative-python-quiz`
3. Run the following command:
   `python run_quiz.py`
4. The quiz window will appear. Answer the questions and see your score at the end.

Technical Documentation

- `quiz_logic.py` : Contains the `Quiz` class for managing questions and scoring.
- `quiz_questions.py` : Stores the quiz questions and answers in a list.

- `quiz_gui.py` : Implements the Tkinter graphical interface (`QuizApp` class).
- `run_quiz.py` : Launches the quiz application.

Developer Manual

- To add or change questions, edit the `quiz_questions` list in `quiz_questions.py` .
- All quiz logic is separated from the interface for easy maintenance and extension.
- You can further extend the app by adding new features or changing the interface in `quiz_gui.py` .

Python Quiz Application: Technical Overview

---

run_quiz.py

Purpose:
This is the main entry point for the application. It initializes the Tkinter root window and launches the GUI by instantiating the `QuizApp` class from `quiz_gui.py` .

```
from quiz_gui import QuizApp
import tkinter as tk

if __name__ == "__main__":
    root = tk.Tk()
    app = QuizApp(root)
    root.mainloop()
```

---

quiz_questions.py

Purpose:
This module contains the data for the quiz. The `quiz_questions` list holds dictionaries, each representing a question and its correct answer. This separation allows for easy modification or extension of the quiz content.

```
# List of quiz questions and answers
quiz_questions = [
    {"question": "What tech uses sound waves to detect objects?",
"answer": "Sonar"},
    {"question": "In pounds, how much did a third-class ticket on the
Titanic cost?", "answer": "7"},
    {"question": "Guernica is a famous painting by which artist?",
"answer": "Picasso"},
    {"question": "What is the capital of Australia?", "answer":
"Canberra"},
    {"question": "What is the largest planet in our solar system?",
```

```
"answer": "Jupiter"},
    {"question": "How many mins are in a half a day?", "answer":
"720"},
    {"question": "What is the chemical symbol for gold?", "answer":
"Au"}
]
```

---

quiz_logic.py

Purpose:

Defines the `Quiz` class, which encapsulates the core quiz logic. It manages the list of questions and the user's score, and provides a method to add new questions.

```python
# Define a Quiz class to manage quiz questions and scoring
class Quiz:
    def __init__(self):
        # Initialize an empty list to store questions and answers
        self.questions = []
        # Initialize the user's score to zero
        self.score = 0

    # Add a question and its answer to the quiz
    def add_question(self, question, answer):
        self.questions.append({"question": question, "answer": answer})
```

---

quiz_gui.py

Purpose:

Implements the graphical user interface using Tkinter. The `QuizApp` class manages the display of questions, user input, and feedback. It interacts with the `Quiz` class for quiz logic and uses the questions from `quiz_questions.py`.

```python
# This file contains the Tkinter interface for the quiz app
 # Import the Tkinter library for GUI
import tkinter as tk
# Import the Quiz logic class
from quiz_logic import Quiz
# Import the list of questions
from quiz_questions import quiz_questions

# Main GUI application class
class QuizApp:
        def __init__(self, master):
                # Store the main window
                self.master = master
                # Set window title
                self.master.title("Welcome to Menna's General Knowledge
Quiz!")
                 # Create a Quiz object to manage questions and scoring

                self.quiz = Quiz()
```

```
                # Add all questions from the imported list to the Quiz
object
                for q in quiz_questions:
                        self.quiz.add_question(q["question"],
q["answer"])

                # Track the current question index
                self.current_question = 0
                # Track the user's score

                self.score = 0
                # Create and pack a label to display the current
question
                self.question_label = tk.Label(master, text="")
                self.question_label.pack()

                # Create and pack an entry widget for user input
                self.answer_entry = tk.Entry(master)
                self.answer_entry.pack()

                # Create and pack a button to submit the answer
                self.submit_btn = tk.Button(master, text="Submit",
command=self.check_answer)
                self.submit_btn.pack()

                # Create and pack a label to display feedback/results
                self.result_label = tk.Label(master, text="")
                self.result_label.pack()

                # Display the first question
                self.next_question()

        def next_question(self):
                # If there are more questions, display the next one
                if self.current_question < len(self.quiz.questions):
                        # Get the question text
                        q = self.quiz.questions[self.current_question]
["question"]
                        # Update the label
                        self.question_label.config(text=q)
                         # Clear the entry box
                        self.answer_entry.delete(0, tk.END)
                         # Clear previous feedback
                        self.result_label.config(text="")
                else:
                        # If no more questions, show the final score
and hide input widgets
                        self.question_label.config(text="Quiz
finished!")
                        # Remove the widget for entering answers
                        self.answer_entry.pack_forget()
                        self.submit_btn.pack_forget()
                        # show the final score
                        self.result_label.config(text=f"Final score:
{self.score}/{len(self.quiz.questions)}")
```

```
        def check_answer(self):
                # Get the user's answer from the entry box
                user_answer = self.answer_entry.get()
                # Get the correct answer for the current question
                correct = self.quiz.questions[self.current_question]
["answer"]
                # Compare answers (case-insensitive)
                if user_answer.strip().lower() == correct.lower():
                        # Increment score if correct
                        self.score += 1
                        # Show positive feedback
                        self.result_label.config(text="Correct!")
                else:
                         # Show correct answer
                        self.result_label.config(text=f"Wrong! Correct:
{correct}")
                          # Move to the next question
                self.current_question += 1
                # Wait 1 second, then show next so that the user has
time to see the feedback
                self.master.after(1000, self.next_question)
```