Python Project 05: Quiz Application

1. Project Overview

In this project, we will build a simple **Quiz Application** using Python and Tkinter.

The application displays multiple-choice questions, allows the user to select an answer, calculates the score, and shows the final result.

Tkinter provides widgets such as buttons and other interface components that can trigger Python functions through callbacks. (Python documentation)

---

2. Features

- Multiple-choice questions
- Four options per question
- Next question button
- Automatic answer checking
- Score calculation
- Question counter
- Final result screen
- Restart quiz
- Modern dark UI

---

3. Technologies

- Python 3
- Tkinter
- Object Oriented Programming

No external packages are required.

---

4. Folder Structure

```
quiz_application/
□
□□□ main.py
□□□ ui.py
□□□ quiz.py
□□□ README.md
```

---

5. How the Quiz Works

The questions are stored in the backend as dictionaries:

```
{
    "question": "Which language is used to create web pages?",
    "options": [
        "Python",
        "HTML",
        "C++",
        "Java"
    ],
    "answer": "HTML"
}
```

The application:

1. Displays a question.
2. Shows four options.

3. User selects an option.
4. The backend checks the answer.
5. Score is updated if the answer is correct.
6. Next question is displayed.
7. Final score is shown after the last question.

---

6. Frontend Code

ui.py

```
import tkinter as tk
from quiz import Quiz



class QuizUI:
    def __init__(self, root):
        self.root = root

        self.root.title("Quiz Application")
        self.root.resizable(False, False)
        self.root.configure(bg="#0f172a")

        self.quiz = Quiz()

        self.selected_answer = tk.StringVar()

        self.create_ui()
        self.show_question()

    def create_ui(self):
        title = tk.Label(
            self.root,
```

```
            text="QUIZ APPLICATION",
            font=("Arial", 24, "bold"),
            fg="#38bdf8",
            bg="#0f172a"
        )
        title.pack(pady=(25, 5))

        self.question_counter = tk.Label(
            self.root,
            text="Question 1 / 5",
            font=("Arial", 11, "bold"),
            fg="#94a3b8",
            bg="#0f172a"
        )
        self.question_counter.pack(pady=(0, 20))

        self.question_label = tk.Label(
            self.root,
            text="",
            font=("Arial", 16, "bold"),
            fg="white",
            bg="#0f172a",
            wraplength=500,
            justify="center"
        )
        self.question_label.pack(
            padx=30,
            pady=(0, 20)
        )

        self.options_frame = tk.Frame(
            self.root,
            bg="#0f172a"
        )
```

```
        self.options_frame.pack(
            padx=30,
            fill="x"
        )

        self.option_buttons = []

        for index in range(4):
            button = tk.Radiobutton(
                self.options_frame,
                text="",
                variable=self.selected_answer,
                value="",
                font=("Arial", 11, "bold"),
                bg="#1e293b",
                fg="white",
                selectcolor="#2563eb",
                activebackground="#334155",
                activeforeground="white",
                anchor="w",
                padx=15,
                pady=12,
                relief="flat",
                cursor="hand2"
            )

            button.pack(
                fill="x",
                pady=5
            )

            self.option_buttons.append(button)

        self.feedback_label = tk.Label(
```

```
            self.root,
            text="",
            font=("Arial", 11, "bold"),
            fg="#f8fafc",
            bg="#0f172a",
            wraplength=450
        )
        self.feedback_label.pack(pady=12)

        self.next_button = tk.Button(
            self.root,
            text="Next Question",
            command=self.next_question,
            font=("Arial", 11, "bold"),
            bg="#2563eb",
            fg="white",
            activebackground="#1d4ed8",
            activeforeground="white",
            relief="flat",
            padx=25,
            pady=9,
            cursor="hand2"
        )
        self.next_button.pack(pady=(5, 10))

        self.score_label = tk.Label(
            self.root,
            text="Score: 0",
            font=("Arial", 11, "bold"),
            fg="#38bdf8",
            bg="#0f172a"
        )
        self.score_label.pack(pady=(0, 25))
```

```
    def show_question(self):
        question = self.quiz.current_question()

        self.selected_answer.set("")
        self.feedback_label.config(text="")

        self.question_counter.config(
            text=(
                f"Question {self.quiz.current_index + 1} "
                f"/ {self.quiz.total_questions()}"
            )
        )

        self.question_label.config(
            text=question["question"]
        )

        for index, button in enumerate(self.option_buttons):
            button.config(
                text=question["options"][index],
                value=question["options"][index],
                state="normal"
            )

        self.next_button.config(
            text="Submit Answer",
            state="normal"
        )

    def next_question(self):
        answer = self.selected_answer.get()

        if not answer:
            self.feedback_label.config(
```

```
                text="Please select an answer.",
                fg="#f87171"
            )
            return

        result = self.quiz.submit_answer(answer)

        if result:
            self.feedback_label.config(
                text="Correct answer!",
                fg="#4ade80"
            )
        else:
            self.feedback_label.config(
                text=(
                    f"Wrong answer. "
                    f"Correct answer: "
                    f"{self.quiz.current_question()['answer']}"
                ),
                fg="#f87171"
            )

        self.score_label.config(
            text=f"Score: {self.quiz.score}"
        )

        for button in self.option_buttons:
            button.config(
                state="disabled"
            )

        if self.quiz.is_finished():
            self.next_button.config(
                text="Show Result",
```

```
            command=self.show_result
        )
    else:
        self.next_button.config(
            text="Next Question",
            command=self.go_to_next
        )

def go_to_next(self):
    self.quiz.next_question()
    self.show_question()

def show_result(self):
    self.question_label.config(
        text=(
            f"Quiz Completed!\n\n"
            f"Your Score: {self.quiz.score} / "
            f"{self.quiz.total_questions()}"
        )
    )

    self.question_counter.config(
        text="Final Result"
    )

    for button in self.option_buttons:
        button.pack_forget()

    self.feedback_label.config(
        text=self.get_result_message(),
        fg="#38bdf8"
    )

    self.next_button.config(
```

```
            text="Restart Quiz",
            command=self.restart
        )

    def get_result_message(self):
        percentage = (
            self.quiz.score /
            self.quiz.total_questions()
        ) * 100

        if percentage >= 80:
            return "Excellent performance!"

        if percentage >= 60:
            return "Good job! Keep practicing."

        return "Keep learning and try again."

    def restart(self):
        for button in self.option_buttons:
            button.pack(
                fill="x",
                pady=5
            )

        self.quiz.reset()
        self.next_button.config(
            command=self.next_question
        )

        self.show_question()
```

---

7. Backend / Quiz Logic Code

quiz.py

```
class Quiz:

    QUESTIONS = [
        {
            "question": "Which language is mainly used to create web page structure?",
            "options": [
                "Python",
                "HTML",
                "C++",
                "Java"
            ],
            "answer": "HTML"
        },
        {
            "question": "Which keyword is used to define a function in Python?",
            "options": [
                "function",
                "define",
                "def",
                "func"
            ],
            "answer": "def"
        },
        {
            "question": "Which data structure stores key-value pairs in Python?",
            "options": [
                "List",
                "Tuple",
                "Dictionary",
                "Set"
```

```
            ],
            "answer": "Dictionary"
        },
        {
            "question": "Which symbol is used for a Python comment?",
            "options": [
                "//",
                "#",
                "<!--",
                "/*"
            ],
            "answer": "#"
        },
        {
            "question": "Which method adds an item to the end of a Python list?",
            "options": [
                "add()",
                "insert()",
                "append()",
                "push()"
            ],
            "answer": "append()"
        }
    ]

    def __init__(self):
        self.reset()

    def reset(self):
        self.questions = [question.copy() for question in self.QUESTIONS]
        self.current_index = 0
        self.score = 0

    def current_question(self):
```

```
        return self.questions[self.current_index]

    def submit_answer(self, answer):
        correct_answer = self.current_question()["answer"]

        if answer == correct_answer:
            self.score += 1
            return True

        return False

    def next_question(self):
        if self.current_index < len(self.questions) - 1:
            self.current_index += 1

    def is_finished(self):
        return self.current_index == len(self.questions) - 1

    def total_questions(self):
        return len(self.questions)
```

---

8. Main File

main.py

```
import tkinter as tk
from ui import QuizUI


def main():
    root = tk.Tk()

    QuizUI(root)
```

```
    root.mainloop()


if __name__ == "__main__":
    main()
```

---

9. How Frontend and Backend Connect

main.py

QuizUI

Quiz

Questions + Answers

Answer Checking

Score

QuizUI

Updated Screen

The **frontend** handles:

- Question display
- Answer selection
- Buttons
- Feedback
- Score display

The **backend** handles:

- Questions
- Correct answers
- Current question
- Answer checking
- Score
- Quiz progress

---

10. How to Run

Open the project folder:

```
cd quiz_application
```

Run:

```
python main.py
```

No external packages are required.

---

11. Basic Testing

Test these cases:

- Question appears correctly.
- Four options are displayed.
- User cannot continue without selecting an answer.

- Correct answer increases the score.
- Wrong answer shows the correct answer.
- Next question loads correctly.
- Final score is calculated.
- Restart Quiz starts from question 1 with score 0.

---

12. Small Improvements

Students can extend the project by adding:

- Question categories
- Difficulty levels
- Timer per question
- Random question order
- Random answer order
- High score
- Question bank using JSON
- Progress bar
- Sound effects
- Images in questions
- Database for storing results

Project Goal

This project practices:

Data → User Input → Validation → Logic → Score → Result

It also introduces a useful real-world pattern: keeping **quiz data and application logic separate from the graphical interface**.

haas.dev

https://dev-roast-app.vercel.app

Visit haas.dev for more resources and guides.