COMP1811 – Python Project Report

Name	Md Sakibuzzaman	Student ID	001173194
Partner's name	Md Mostafizur Rahman Badhon	Partner SIDs	001134460

1. Brief statement of features you have completed

(THIS SECTION SHOULD BE THE SAME FOR BOTH PARTNERS)
Indicate the feature each partner implemented by replacing "developed by" in red below with partner name.

1.1 Circle the parts of the coursework you have **fully completed and are fully working**. Please be accurate.	**Features** *[Developed by]* **F1:** i ☒ ii ☒ iii ☒ *[Developed by]* **F2:** i ☒ ii ☒ iii ☒
1.2 Circle the parts of the coursework you have **partly completed or are partly working.**	**Features** **F1:** i ☒ ii ☒ iii ☒ **F2:** i ☒ ii ☒ iii ☒
Briefly explain your answer if you circled any parts in 1.2	

2. Concise List of Bugs and Weaknesses

2.1 Bugs

While managing or deleting the files, all the data present in the file is complete removed to again reuse the past data, it needs to manually copy externally in JSON format to be able to work properly in the solution.

2.2 Weaknesses

The data is stored in the text file formatting manner. However, the data format stored in the text file is that of the JSON file format. It was done for the ease of question handling in MCQ quiz sections.

3. Description of the features implemented

- The user should be able to add or manage new modules
- The user should be able to give MCQ quiz
- The user will be able to view current quiz result data
- The user will be able to check the time stamped quiz result in result.txt file
- The user will be able to use the quiz in beautifully structured GUI interface
- The user will be able to exit from the GUI interface by selecting quit button

4. Classes and OOP Features

*List all the classes used in your program and include the attributes and behaviours for each. You may use a class diagram to illustrate these classes. Your narrative for section 3.2 should describe the design decisions you made and the OOP techniques used. Each partner must list the classes they developed separately and provide an exposition on the choice of classes, class design and OOP features implemented. (**200-400 words for each partner**). (THIS SECTION SHOULD BE THE SAME FOR BOTH PARTNERS)*

3.1 Classes Used

In the current section, two classes were primarily used They are class Quiz and class record writer.

3.2 Brief Explanation of Class Design and OOP Features Used

Class record writer is used to work with the result writing section of the project where the program will note the result of each quiz in time stamped manner.

Class Quiz will handle all the functions of the project including GUI implementation, and handling the following features:

- add or manage new modules
- Start and condunt MCQ quiz
- Update and view current quiz result data
- Store time stamped quiz result in result.txt file
- exit from the GUI interface by selecting quit button

5. Code for the Classes Created

*Add the **code for each of the classes you have implemented yourself** here. If you have contributed to parts of classes, please highlight those parts in a different colour. Copy and paste relevant code - actual code please, no screenshots! Make it easy for the tutor to read. Add explanation if necessary – though your in-code comments should be clear enough. You will lose marks if screenshots are provided instead of code.*
(COMPLETE THIS SECTION INDIVIDUALLY – only list the code for the classes you developed individually. DO NOT provide a listing of the entire code. You will be marked down if a full code listing is provided.)

4.1 Class Quiz

```
from tkinter import *

from tkinter import messagebox as mb

from record_writer import *

import json


class Quiz:

    def __init__(self):

        self.q_no = 0

        self.display_title()
        self.display_question()

        self.opt_selected = IntVar()

        self.opts = self.radio_buttons()

        self.display_options()

        self.buttons()

        self.data_size = len(question)

        self.correct = 0

    def display_result(self):
        # calculates the wrong count
        wrong_count = self.data_size - self.correct
        correct = f"Correct: {self.correct}"
        wrong = f"Wrong: {wrong_count}"

        score = int(self.correct / self.data_size * 100)
        result = f"Score: {score}%"

        mb.showinfo("Result", f"{result}\n{correct}\n{wrong}")
        record = "Result " + f"{result}\n{correct}\n{wrong}"
        print(record)
        write = record_writer()
        write.write_file("result.txt", record)
        print()

    def check_ans(self, q_no):
        if self.opt_selected.get() == answer[q_no]:
            return True

    def next_btn(self):

        if self.check_ans(self.q_no):
            self.correct += 1

        self.q_no += 1

        if self.q_no == self.data_size:

            self.display_result()

            gui.destroy()
        else:

            self.display_question()
            self.display_options()

    def module_btn(self):
        my_file = open("modules.txt", "a")

        mod = input("Enter a module name to add: ")
        my_file.write(mod + "\n")
        # file read
        my_file = open("modules.txt")
        content = my_file.read()

        my_file.close()
        print("All the module details are:")
        print("--------")
        print(content)
        print("--------")

    def question_btn(self):
        print("question_btn")
        input("Press Enter")

    def module_del_btn(self):
        print("module_del_btn")
        input("Press Enter")

    def question_del_btn(self):
        print("module_del_btn")
        input("Press Enter")

    def buttons(self):

        module_button = Button(gui, text="AddNew Module", command=self.module_btn,
                               width=13, bg="gray", fg="white", font=("ariel", 16, " bold"))

        module_button.place(x=100, y=50)

        module_del_button = Button(gui, text="Manage/Del Module",
command=self.module_del_btn,
                               width=17, bg="gray", fg="white", font=("ariel", 16, " bold"))

        module_del_button.place(x=100, y=100)

        question_button = Button(gui, text="Manage Question", command=self.question_btn,
                                width=15, bg="gray", fg="white", font=("ariel", 16, " bold"))

        question_button.place(x=350, y=50)

        question_del_button = Button(gui, text="Manage/Del Question",
command=self.question_del_btn,
                               width=19, bg="gray", fg="white", font=("ariel", 16, " bold"))

        question_del_button.place(x=350, y=100)
        next_button = Button(gui, text="Next", command=self.next_btn,
                             width=10, bg="#87ceeb", fg="white", font=("ariel", 16, "bold"))

        next_button.place(x=350, y=380)

        quit_button = Button(gui, text="Quit", command=gui.destroy,
                             width=5, bg="gray", fg="white", font=("ariel", 16, " bold"))

        quit_button.place(x=700, y=50)

    def display_options(self):
        val = 0

        self.opt_selected.set(0)

        for option in options[self.q_no]:
            self.opts[val]['text'] = option
            val += 1

    def display_question(self):

        q_no = Label(gui, text=question[self.q_no], width=60,
                     font=('ariel', 16, 'bold'), anchor='w')

        q_no.place(x=70, y=150)

    def display_title(self):

        # The title to be shown
        title = Label(gui, text="GUI QUIZ",
                      width=50, bg="gray", fg="white", font=("ariel", 20, "bold"))

        title.place(x=0, y=2)

    def radio_buttons(self):

        q_list = []

        y_pos = 200

        while len(q_list) < 4:
            radio_btn = Radiobutton(gui, text=" ", variable=self.opt_selected,
                                    value=len(q_list) + 1, font=("ariel", 14))
            q_list.append(radio_btn)

            radio_btn.place(x=100, y=y_pos)

            y_pos += 40

        return q_list


gui = Tk()

gui.geometry("800x450")

gui.title("GUI Quiz")

with open('questions.txt') as f:
    data = json.load(f)

question = (data['question'])
options = (data['options'])
answer = (data['answer'])

quiz = Quiz()
print("")
input("Hello, Press Enter to start the quiz!")

gui.mainloop()
```

4.2 CLASS RECORD_WRITER

```
import datetime


class record_writer:
    def write_file(file, records):
        f = open(file, "a")
        current = datetime.datetime.now()
        date_time = current.strftime("%m/%d/%Y, %H:%M:%S")
        f.write("Record date/time: "+date_time + "\n")
        f.write(records)
        f.write("\n-----\n")
        f.close()
```

6. Testing

Describe the process you took to test your code and to make sure the program functions as required. Provide the detailed test plan used. Also, indicate the testing you did after integrating your code with your partner's. (COMPLETE THIS SECTION INDIVIDUALLY)

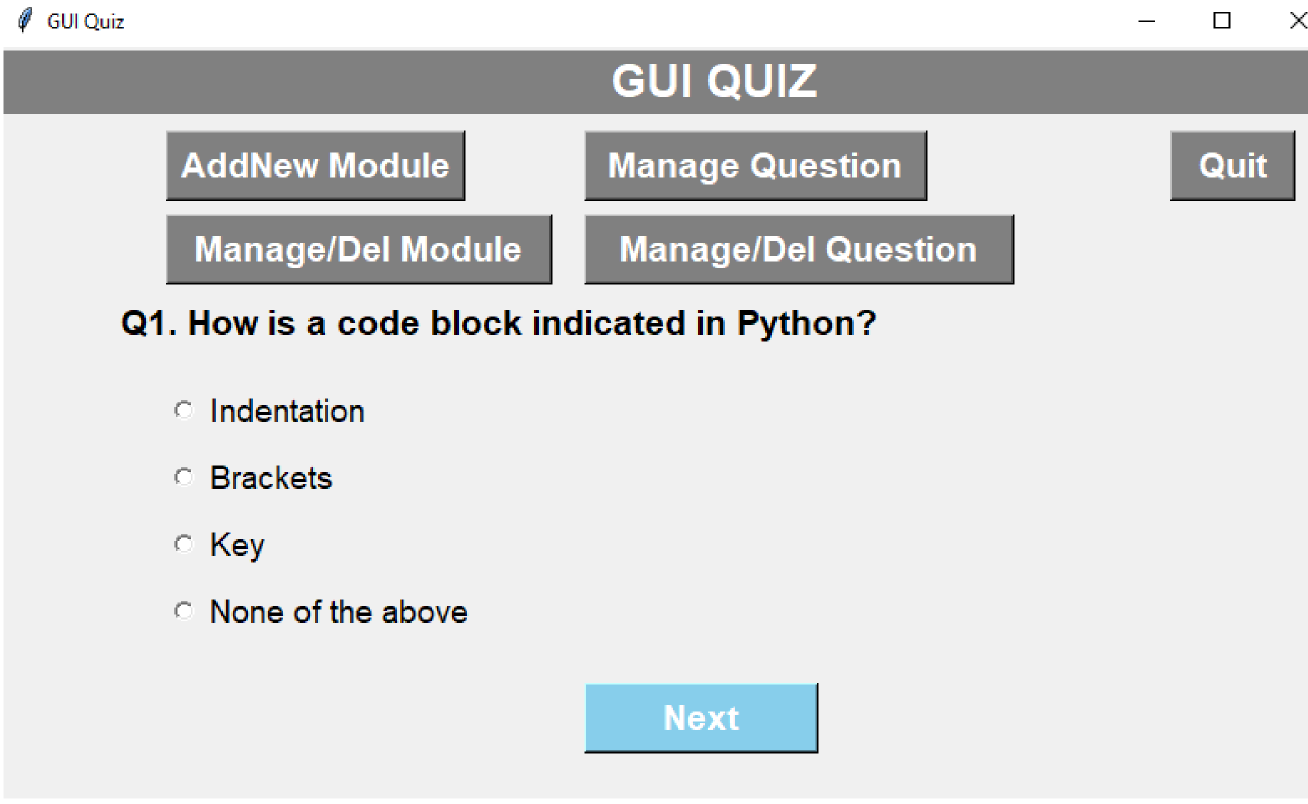
The testing was done correctly. And I have added a video file in the zip folder to demonstrate the testing. As far as my testing goes everything works perfectly.

7. Annotated Screenshots Demonstrating Implementation

Provide screenshots that demonstrate the features implemented. Annotate each screenshot and if necessary, provide a brief description for ***each*** ***(up to 100 words****) to explain the code in action. Make sure the screenshots make clear what you have implemented and achieved.*
(THIS SECTION SHOULD BE THE SAME FOR BOTH PARTNERS)

7.1 Feature F1

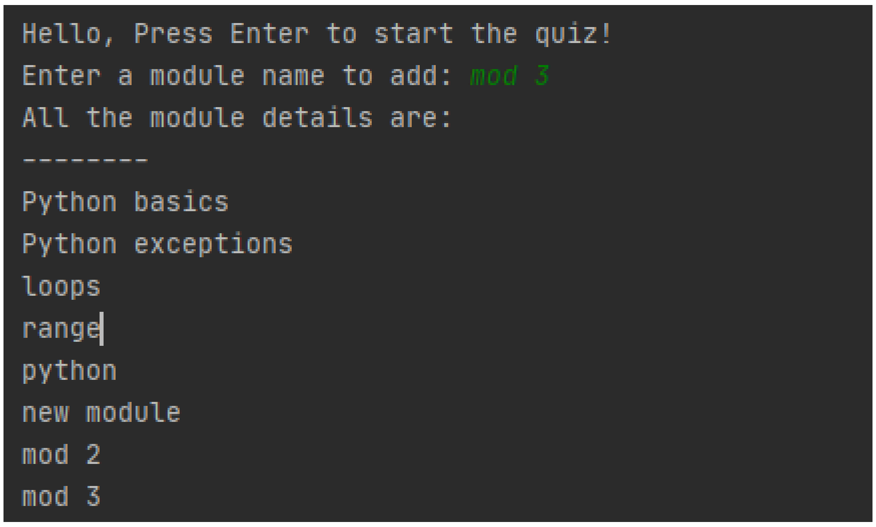
a. Sub-feature I- screenshots



Q1. How is a code block indicated in Python?

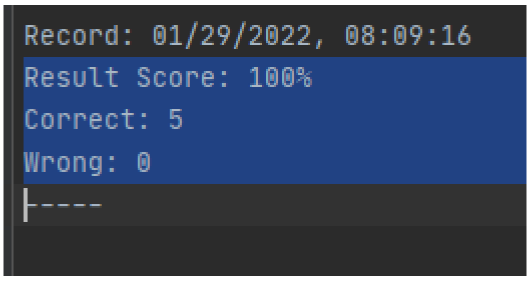
- Indentation
- Brackets
- Key
- None of the above

b. Sub-feature II- screenshots



```
Hello, Press Enter to start the quiz!
Enter a module name to add: mod 3
All the module details are:
--------
Python basics
Python exceptions
loops
range
python
new module
mod 2
mod 3
```

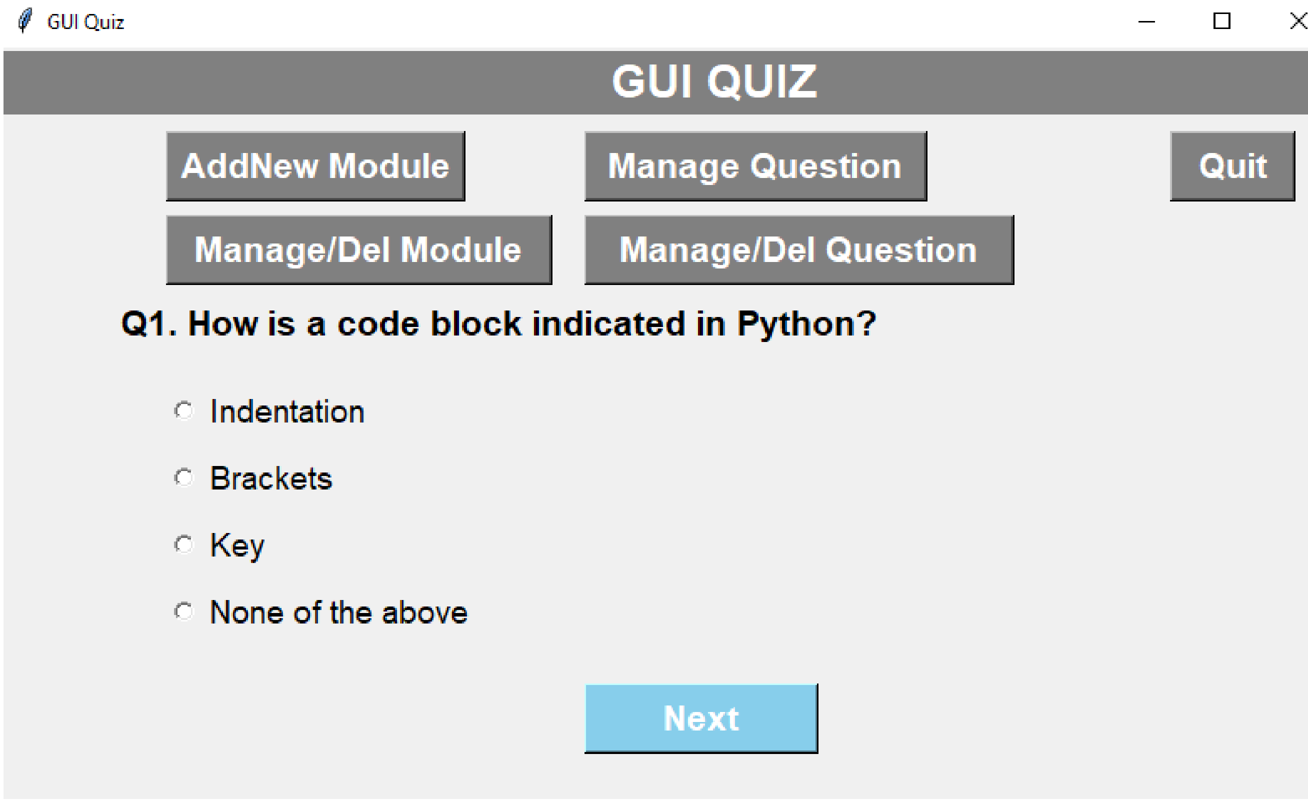
c. SUB-FEATURE III- SCREENSHOTS



```
Record: 01/29/2022, 08:09:16
Result Score: 100%
Correct: 5
Wrong: 0
-----
```

7.2 FEATURE F2 (*indicate name of partner that developed feature here*)

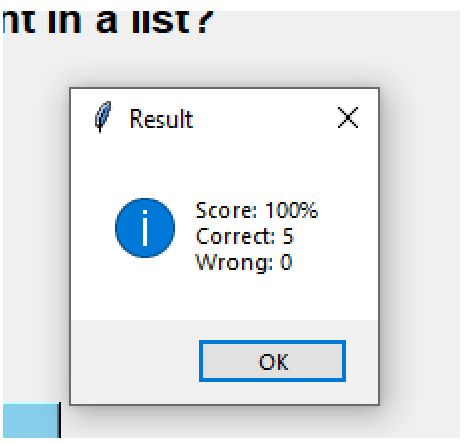
a. SUB-FEATURE I- SCREENSHOTS ...



Q1. How is a code block indicated in Python?

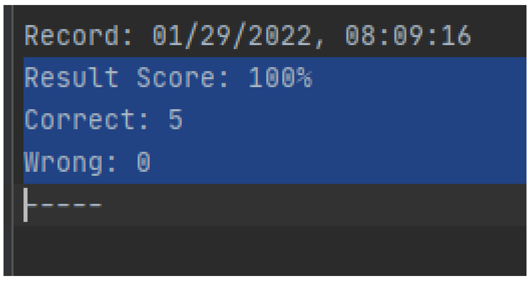
- Indentation
- Brackets
- Key
- None of the above

b. SUB-FEATURE II- SCREENSHOTS …



Score: 100%
Correct: 5
Wrong: 0

c. SUB-FEATURE III- SCREENSHOTS



```
Record: 01/29/2022, 08:09:16
Result Score: 100%
Correct: 5
Wrong: 0
-----
```

8. Evaluation

Give a reflective, critical self-evaluation of your experience developing the project and discuss what you would do if you had more time to work on the project. Answer the following questions for the reflection and write ***350-400 words overall****. Please include an actual word count for this section.*
(COMPLETE THIS SECTION INDIVIDUALLY)

8.1 Evaluate how well your design and implementation meet the requirements

The design and implementation was overall good

8.2 Evaluate you own and your group's performance

The implementation was overall good

8.2.1 What went well?

GUI design

8.2.2 What went less well?

Data conversion and handling

8.2.3 What was learnt?

OOP features and data handling

8.2.4 How would a similar task be completed differently?

By using database instead of text file system in the solution.

8.2.5 How could the module be improved?

Incorporating for both text base and GUI based features as per the user criterion

8.3 Self-assessment

Please assess yourself objectively for each section shown below and then enter the total mark you expect to get. Marks for each assessment criteria are indicated between parentheses.

Code development (70)

Features Implemented [30]

Sub-feature i (up to 8)

Sub-features have not been implemented – 0
Attempted, not complete or very buggy – 1 or 2
Implemented and functioning without errors but not integrated – 3 or 4
Implemented and fully integrated but buggy – 5 or 6
Implemented, fully integrated and functioning without errors – 7 or 8

Sub-feature ii (up to 10)

Sub-features have not been implemented – 0
Attempted, not complete or very buggy – 1 or 2
Implemented and functioning without errors but not integrated – 3 to 5
Implemented and fully integrated but buggy – 6 to 8
Implemented, fully integrated and functioning without errors – 9 or 10

Sub-feature iii (up to 12)

Sub-features has not been implemented – 0
Attempted, not complete or very buggy – 1 to 3
Implemented and functioning without errors but not integrated – 4 to 6
Implemented and fully integrated but buggy – 7 to 9
Implemented, fully integrated and functioning without errors – 10 to 12

For this criterion I think I got: 20 out of 30

Use of OOP techniques [25]

Abstraction (up to 10)

No classes have been created – 0
Classes have been created superficially and not instantiated or used – 1 or 2
Classes have been created but only some have been instantiated and used – 3 or 4
Useful classes and objects have been created and used correctly – 5 to 7
The use of classes and objects exceeds the specification – 8 to 10

Encapsulation (up to 10)

No encapsulation has been used – 0
Class variables and methods have been encapsulated superficially – 1 to 3
Class variables and methods have been encapsulated correctly – 4 to 6
The use of encapsulation exceeds the specification – 7 to 10

Inheritance (up to 5)

No inheritance has been used – 0
Classes have been inherited superficially – 1
Classes have been inherited correctly – 2 to 4
The use of inheritance exceeds the specification – 5

Bonus marks will be awarded for the appropriate use of polymorphism (bonus marks up to 10)

For this criterion I think I got: 20 out of 25

Quality of Code [15]

Code Duplication (up to 8)

Code contains too many unnecessary code repetition – 0
Regular occurrences of duplicate code – 1 to 3
Occasional duplicate code – 4 to 5
Very little duplicate code – 6 to 7
No duplicate code – 8

PEP8 Conventions and naming of variables, methods and classes (up to 4)

PEP8 and naming convention has not been used – 0
PEP8 and naming convention has been used occasionally – 1
PEP8 and naming convention has been used, but not regularly – 2
PEP8 and naming convention has been used regularly – 3
PEP8 convention used professionally and all items have been named correctly – 4

In-code Comments (up to 3)

No in-code comments – 0
Code contains occasional in-code comments – 1
Code contains useful and regular in-code comments – 2
Thoroughly commented, good use of docstrings, and header comments describing.py files – 3

For this criterion I think I got: 15 out of 15

Documentation (20)

Design (up to 10) clear exposition about the design and decisions for OOP use

The documentation cannot be understood on first reading or mostly incomplete – 0
The documentation is readable, but a section(s) are missing – 1 to 3
The documentation is complete – 4 to 6
The documentation is complete and of a high standard – 7 to 10

Testing (5)

Testing has not been demonstrated in the documentation – 0
Little white box testing has been documented – 1 or 2
White box testing has been documented for all the coursework – 3 or 4
White box testing has been documented for the whole system – 5

Evaluation (5)

No evaluation was shown in the documentation – 0
The evaluation shows a lack of thought – 1 or 2
The evaluation shows thought – 3 or 4
The evaluation shows clear introspection, demonstrates increased awareness – 5

For this criterion I think I got: 15 out of 20

Acceptance Tests - Demonstrations (10)

Final Demo (up to 10)

Not attended or no work demonstrated – 0
Work demonstrated was not up to the standard expected – 1 to 3
Work demonstrated was up to the standard expected – 4 to 7

Work demonstrated exceeded the standard expected – 8 to 10

For this criterion I think I got: 10 out of 10

I think my overall mark would be: 80 out of 100

9. GROUP PRO FORMA

Describe the division of work and agree percentage contributions. The pro forma must be signed by all group members and an identical copy provided in each report. If you cannot agree percentage contributions, please indicate so in the notes column and provide your reasoning.
(THIS SECTION SHOULD BE THE SAME FOR BOTH PARTNERS)

Partner ID	Tasks/Features Completed	%Contribution	Signature	Notes
1	**001173194**	50%	Md Sakibuzzaman	
2	001134460	50%	Md Mostafizur Rahman Badhon	
	Total	100%		

APPENDIX A: CODE LISTING

*Provide a complete listing of all the *.py files in your PyCharm project. Make sure your code is well commented and applies professional Python convention (refer to PEP 8 for details). The code listed here must match that uploaded to Moodle. Please copy and paste the actual code – no screenshots please! You will lose marks if screenshots are provided instead of code.*
(THIS SECTION SHOULD BE THE SAME FOR BOTH PARTNERS)

Quiz Codes:

```
from tkinter import *

from tkinter import messagebox as mb

from record_writer import *

import json


class Quiz:

    def __init__(self):

        self.q_no = 0

        self.display_title()
        self.display_question()

        self.opt_selected = IntVar()

        self.opts = self.radio_buttons()

        self.display_options()

        self.buttons()

        self.data_size = len(question)

        self.correct = 0

    def display_result(self):

        # calculates the wrong count
        wrong_count = self.data_size - self.correct
        correct = f"Correct: {self.correct}"
        wrong = f"Wrong: {wrong_count}"

        score = int(self.correct / self.data_size * 100)
        result = f"Score: {score}%"

        mb.showinfo("Result", f"{result}\n{correct}\n{wrong}")
        record = "Result " + f"{result}\n{correct}\n{wrong}"
        print(record)
        write = record_writer()
        write.write_file("result.txt", record)
        print()
    def check_ans(self, q_no):
        if self.opt_selected.get() == answer[q_no]:
            return True

    def next_btn(self):

        if self.check_ans(self.q_no):
            self.correct += 1

        self.q_no += 1

        if self.q_no == self.data_size:

            self.display_result()

            gui.destroy()
        else:

            self.display_question()
            self.display_options()

    def module_btn(self):
        my_file = open("modules.txt", "a")

        mod = input("Enter a module name to add: ")
        my_file.write(mod + "\n")
        # file read
        my_file = open("modules.txt")

        content = my_file.read()

        my_file.close()
        print("All the module details are:")
        print("---------")
        print(content)
        print("---------")

    def question_btn(self):
        print("question_btn")
        input("Press Enter")

    def module_del_btn(self):
        print("module_del_btn")
        input("Press Enter")

    def question_del_btn(self):
        print("module_del_btn")
        input("Press Enter")

    def buttons(self):

        module_button = Button(gui, text="AddNew Module", command=self.module_btn,
                               width=13, bg="gray", fg="white", font=("ariel", 16,
" bold"))

        module_button.place(x=100, y=50)

        module_del_button = Button(gui, text="Manage/Del Module",
command=self.module_del_btn,
                               width=17, bg="gray", fg="white", font=("ariel", 16,
" bold"))
```

```
        module_del_button.place(x=100, y=100)

        question_button = Button(gui, text="Manage Question",
command=self.question_btn,
                                 width=15, bg="gray", fg="white", font=("ariel",
16, " bold"))

        question_button.place(x=350, y=50)

        question_del_button = Button(gui, text="Manage/Del Question",
command=self.question_del_btn,
                               width=19, bg="gray", fg="white", font=("ariel", 16,
" bold"))

        question_del_button.place(x=350, y=100)

        next_button = Button(gui, text="Next", command=self.next_btn,
                             width=10, bg="#87ceeb", fg="white", font=("ariel",
16, "bold"))

        next_button.place(x=350, y=380)

        quit_button = Button(gui, text="Quit", command=gui.destroy,
                             width=5, bg="gray", fg="white", font=("ariel", 16, "
bold"))

        quit_button.place(x=700, y=50)

    def display_options(self):
        val = 0

        self.opt_selected.set(0)

        for option in options[self.q_no]:
            self.opts[val]['text'] = option
            val += 1

    def display_question(self):

        q_no = Label(gui, text=question[self.q_no], width=60,
                     font=('ariel', 16, 'bold'), anchor='w')

        q_no.place(x=70, y=150)

    def display_title(self):

        # The title to be shown
        title = Label(gui, text="GUI QUIZ",
                      width=50, bg="gray", fg="white", font=("ariel", 20, "bold"))

        title.place(x=0, y=2)

    def radio_buttons(self):

        q_list = []

        y_pos = 200

        while len(q_list) < 4:
            radio_btn = Radiobutton(gui, text=" ", variable=self.opt_selected,
                                    value=len(q_list) + 1, font=("ariel", 14))
            q_list.append(radio_btn)

            radio_btn.place(x=100, y=y_pos)

            y_pos += 40

        return q_list


gui = Tk()

gui.geometry("800x450")

gui.title("GUI Quiz")

with open('questions.txt') as f:
    data = json.load(f)

question = (data['question'])
options = (data['options'])
answer = (data['answer'])

quiz = Quiz()
print("")
input("Hello, Press Enter to start the quiz!")

gui.mainloop()
```

Record Writer Code:

```
import datetime




class record_writer:

  def write_file(file, records):

    f = open(file, "a")

    current = datetime.datetime.now()

    date_time = current.strftime("%m/%d/%Y, %H:%M:%S")

    f.write("Record date/time: "+date_time + "\n")

    f.write(records)

    f.write("\n-----\n")



    f.close()
```