ENSAM
ÉCOLE NATIONALE SUPÉRIEURE D'ARTS ET MÉTIERS
UNIVERSITÉ HASSAN II DE CASABLANCA

RAPPORT

PROJET PYTHON

1ère année du cycle d'ingénieur
IAGI

Réalisé par :
CHAKIR ACHRAF
RESSAKI DOUHA
HANAFI RIHAB

Encadré par :
Pr. HAIN MUSTAPHA

Introduction

L'application développée en Python constitue un outil interactif de questionnaire avec des fonctionnalités avancées d'authentification et de suivi des scores. En combinant les bibliothèques Tkinter pour l'interface graphique et SQLite3 pour la gestion de base de données, l'application offre une expérience utilisateur fluide et sécurisée.
Elle se compose de deux pages principales : la page de connexion et la page du questionnaire. La première offre une interface conviviale permettant à l'utilisateur de saisir son nom d'utilisateur, de se connecter, d'afficher les scores enregistrés de différents utilisateurs, ou de quitter l'application. La seconde page, quant à elle, propose un questionnaire interactif avec des questions à choix multiples

- **Fonctionnalités da la page connexion :**

Lorsque l'application démarre, une fenêtre de connexion s'affiche. Cette interface élémentaire propose un champ de saisie pour le nom d'utilisateur. L'utilisateur peut saisir son nom d'utilisateur, puis cliquer sur le bouton "Se Connecter". Si un nom d'utilisateur est saisi, la page du questionnaire s'affiche.
Un bouton **"Afficher les scores"** permet à l'utilisateur de consulter les scores enregistrés dans la base de données. Si aucun score n'est enregistré, une boîte de dialogue informative apparaît.
Et finalement un bouton "Quitter" qui ferme l'application.

- **Fonctionnalités de la page du questionnaire :**

La page du questionnaire est présentée dès qu'un utilisateur se connecte avec succès. Elle affiche une série de questions à choix multiples, chacune accompagnée de plusieurs options. L'utilisateur peut sélectionner ses réponses en utilisant les boutons radio associés à chaque option.
Un bouton **"Afficher le résultat"** permet à l'utilisateur de voir son score après avoir répondu à toutes les questions.

• **Fonctionnalités de la page du questionnaire :**

Ce score affiché est le score total par rapport au nombre total de questions. Si une réponse est manquante, une boîte de dialogue avertit l'utilisateur de la nécessité de répondre à toutes les questions avant de pouvoir afficher les résultats.

Un bouton "Effacer les réponses" permet de réinitialiser toutes les réponses sélectionnées, permettant à l'utilisateur de recommencer le questionnaire.

Enfin, un bouton "Revenir au menu" ferme la page du questionnaire, enregistre le score dans la base de données, et ramène l'utilisateur à la page de connexion.

• Explication du code:

```
1  import tkinter as tk
2  from tkinter import messagebox
3  import sqlite3
```

Importation du module tkinter le renomme en tk pour faciliter l'accès.
Importation de la classe messagebox de Tkinter pour afficher des boîtes de dialogue.
Importation du module sqlite3:

```
5  class LoginPage:
6      def __init__(self, window):
7          self.window = window
8          self.window.title("Page de Connexion")
9
10         self.username_label = tk.Label(window, text="Nom d'utilisateur:")
11         self.username_label.grid(row=0, column=0, pady=10)
12         self.username_entry = tk.Entry(window)
13         self.username_entry.grid(row=0, column=1, pady=10)
14
15         login_button = tk.Button(window, text="Se Connecter", command=self.login)
16         login_button.grid(row=2, column=0, columnspan=2, pady=10)
17
18         scores_button = tk.Button(window, text="Afficher les scores", command=self.show_scores)
19         scores_button.grid(row=3, column=0, columnspan=2, pady=10)
20
21         quit_button = tk.Button(window, text="Quitter", command=window.quit)
22         quit_button.grid(row=4, column=0, columnspan=2, pady=10)
23
```

- On définie la classe LoginPage pour créer la page de connexion

__init__(self, window): Le constructeur de la classe en paramètre la fenêtre. Il permet de crée des éléments tels que des étiquettes, des champs d'entrée et des boutons pour la page de connexion.

login_button=tk.Button(window,text="SeConnecter",command=self.login permet de crée le bouton du login, en cliquant sur le bouton on fait appelle à la methode login

scores_button = tk.Button(window, text="Afficher les scores", command=self.show_scores) : permet de crée le bouton qui permet d'afficher le score, en cliquant sur le bouton on fait appelle à la methode show.scores

quit_button = tk.Button(window, text="Quitter", command=window.quit) : permet de crée le bouton quit afin de fermer la fenêtre

```
24      def login(self):
25          if self.username_entry.get():
26              self.show_questionnaire(self.username_entry.get())
27
```

Methode qui Vérifie si le champ d'entrée du nom d'utilisateur n'est pas vide et appelle self.show_questionnaire avec le nom d'utilisateur en argument.

```
28      def show_scores(self):
29          db_file = "questionnaire_responses.db"
30          connection = sqlite3.connect(db_file)
31          cursor = connection.cursor()
32          cursor.execute('CREATE TABLE IF NOT EXISTS user_scores (username TEXT PRIMARY KEY, score INTEGER)')
33          cursor.execute('SELECT username, score FROM user_scores')
34          scores = cursor.fetchall()
35
36          if not scores:
37              messagebox.showinfo("Scores", "Aucun score enregistré.")
38          else:
39              score_str = "\n".join(f" {username} : {score}" for username, score in scores)
40              messagebox.showinfo("Scores", f"Scores : \n {score_str}")
41
42          connection.close()
43
```

Méthode show_scores : Ouvre la base de données, crée la table "user_scores" si elle n'existe pas, et récupère les scores des utilisateurs. Affiche les scores dans une boîte de dialogue si, ils existent sinon elle affiche un warning.

```
43
44      def show_questionnaire(self, username):
45          self.window.withdraw()
46          questionnaire_window = tk.Toplevel(self.window)
47          QuestionnairePage(questionnaire_window, "questionnaire_responses.db", username, self.window)
48
```

Méthode show_questionnaire : Masque la fenêtre actuelle (withdraw), en utilisant tk.Toplevel on peut créer une fenêtre supplémentaire (ou enfant) indépendante de la fenêtre principale. Cette nouvelle fenêtre est stockée dans la variable questionnaire_window et instancie la classe QuestionnairePage pour afficher le questionnaire.

```python
class QuestionnairePage:
    def __init__(self, window, db_file, username, main_window):
        self.window = window
        self.window.title("Questionnaire")

        self.main_window = main_window

        self.connection = sqlite3.connect(db_file)
        self.create_table()

        self.username = username

        self.questions = [
            Question("Question 1: Python est un langage _____?", ["interprété", "machine", "binaire"], 0),
            Question("Question 2: Laquelle des fonctions suivantes convertit un « string » en « float » en python?", ["str(x)", "float(x)", "long(x [,base] )"], 1),
            Question("Question 3: En Python, laquelle des fonctions suivantes vérifie dans une chaîne de caractères que tous les caractères sont des chiffres?", ["isdigit()", "capitalize()", "isalnum()"], 0),
            Question("Question 4: Quelle est la fonction qui compare les éléments des deux listes?", ["eq(list1, list2)", "cmp(list1, list2)", "max(list1, list2)"], 1),
            Question("Question 5: Laquelle des fonctions suivantes renvoie le plus petit caractère de la chaîne str?", ["lower()", " lstrip()", "min(str)"], 2),
            Question("Question 6: Quelle fonction est utilisée pour ouvrir le fichier en lecture en Python?", ["fopen(file_name, mode)", "open(file_name, mode)", "open_file(file_name, mode)"], 1),
            Question("Question 7: En python, quel mot clé est utilisé pour commencer une fonction?", ["function", "def", "import"], 1),
            Question("Question 8: Quelle sera la sortie du code suivant? chr(ord('A'))", ["65", "a", "A"], 0),
            Question("Question 9: Lorsqu'une fonction est définie dans une classe, on l'appelle _____?", ["Module", "Classe", "Méthode"],2),
            Question("Question 10: Quelle sera la sortie du code suivant? [ (a,b) for a in range(3) for b in range(a) ]", ["[(1,0),(2,0),(2,1)]", " [(1,0),(2,1),(2,1)]", "[(0,0),(1,1),(2,2)]"], 0)
        ]

        self.labels = []
        self.radio_vars = []

        for i, question in enumerate(self.questions):
            label = tk.Label(window, text=question.text)
            label.grid(row=i, column=0, sticky="w")
            self.labels.append(label)

            radio_var = tk.IntVar()
            radio_var.set(-1)  # Initialiser avec la valeur par défaut -1
            self.radio_vars.append(radio_var)

            for j, option in enumerate(question.options):
                radio_button = tk.Radiobutton(window, text=option, variable=radio_var, value=j)
                radio_button.grid(row=i, column=j + 1)

        result_button = tk.Button(window, text="Afficher le résultat", command=self.show_score)
        result_button.grid(row=len(self.questions), column=0, pady=10)

        clear_button = tk.Button(window, text="Effacer les réponses", command=self.clear_answers)
        clear_button.grid(row=len(self.questions) + 1, column=0, pady=10)

        back_to_menu_button = tk.Button(window, text="Revenir au menu", command=self.back_to_menu)
        back_to_menu_button.grid(row=len(self.questions) + 2, column=0, pady=10)
```

Définition de la classe QuestionnairePage :
__init__(self, window, db_file, username, main_window): Le constructeur prend la fenêtre du questionnaire, le fichier de base de données, le nom d'utilisateur, et la fenêtre principale en paramètres.
self.create_table() : fait appelle à la methode create_table
self.question = [une liste des instances de la classes Question]
les deux boucles imbriquées permet de crée les étiquettes et les boutons radio qui représente les différentes propositions dans chaque question.
result_button = tk.Button(window, text="Afficher le résultat", command=self.show_score) : permet de créer un bouton pour afficher les scores
clear_button = tk.Button(window, text="Effacer les réponses", command=self.clear_answers) : permet de créer un bouton afin de désélectionner les réponses cocher
back_to_menu_button = tk.Button(window, text="Revenir au menu", command=self.back_to_menu) : permet de créer un bouton afin de retourner à la page de login

```python
    def create_table(self):
        cursor = self.connection.cursor()
        cursor.execute('CREATE TABLE IF NOT EXISTS user_scores (username TEXT PRIMARY KEY, score INTEGER)')
        self.connection.commit()
```

Méthode create_table : Crée la table "user_scores" dans la base de données si elle n'existe pas.

```
104     def insert_user_score(self, username, score):
105         cursor = self.connection.cursor()
106         cursor.execute('INSERT OR REPLACE INTO user_scores VALUES (?, ?)', (username, score))
107         self.connection.commit()
108
```

Méthode insert_user_score : Insère ou met à jour le score de l'utilisateur dans la table de la base de donnée

```
def show_score(self):
    # Vérifier si toutes les réponses ont été sélectionnées
    if -1 in [var.get() for var in self.radio_vars]:
        messagebox.showwarning("Attention", "Veuillez répondre à toutes les questions.")
        return

    # Calculer le score
    score = sum(question.is_correct(var.get()) for question, var in zip(self.questions, self.radio_vars))

    # Afficher le score
    result_label = tk.Label(self.window, text=f"Votre score : {score}/{len(self.questions)}")
    result_label.grid()

    # Insérer ou mettre à jour le score de l'utilisateur dans la base de données
    self.insert_user_score(self.username, score)

def clear_answers(self):
    # Clear radio button selections
    for radio_var in self.radio_vars:
        radio_var.set(-1)  # Set default value to -1

def back_to_menu(self):
    self.connection.close()
    self.window.destroy()
    self.main_window.deiconify()
```

Méthode show_score : Vérifie si toutes les réponses ont été sélectionnées, calcule le score, l'affiche dans la fenêtre, et insère ou met à jour le score dans la base de données.

```
125     def clear_answers(self):
126         # Clear radio button selections
127         for radio_var in self.radio_vars:
128             radio_var.set(-1)  # Set default value to -1
129
```

Méthode clear_answers : permet de désélectionner les boutons radio cocher

```
130     def back_to_menu(self):
131         self.connection.close()
132         self.window.destroy()
133         self.main_window.deiconify()
134
```

Méthode back_to_menu : permet de retourner à la page de login en fermant la fenêtre actuelle.

```python
class Question:
    def __init__(self, text, options, correct_answer):
        self.text = text
        self.options = options
        self.correct_answer = correct_answer

    def is_correct(self, user_answer):
        if user_answer == -1:
            return False
        return user_answer == self.correct_answer
```

Définition de la class Question :
def __init__(self, text, options, correct_answer) : Le constructeur initialise un objet Question avec un texte de question (text), une liste d'options (options), et l'index de la réponse correcte (correct_answer).
Méthode is_correct(self, user_answer): prend une réponse d'utilisateur (user_answer) en paramètre. Vérifie si la réponse est correcte en comparant avec l'index de la réponse correcte. Si user_answer est égal à -1 (signifiant que l'utilisateur n'a pas répondu), la méthode retourne False.

```python
if __name__ == "__main__":
    window = tk.Tk()
    LoginPage(window)
    window.mainloop()
```

Cette partie du code est un bloc qui s'exécute uniquement si le script est exécuté en tant que programme principal.

Conclusion

En conclusion, cette Application de Quiz met en valeur la puissance de Python et de la bibliothèque tkinter pour développer une interface utilisateur interactive. De la page de connexion conviviale au questionnaire dynamique, le projet offre une expérience intuitive tant pour les développeurs que pour les utilisateurs. L'intégration de SQLite pour le suivi des scores ajoute une couche de persistance, permettant aux utilisateurs de suivre leur progression au fil du temps.