Bing

Oups, je suis désolé. J'ai oublié d'afficher le code. Voici le code que j'ai rédigé pour vous. 🙏

# Importer les bibliothèques nécessaires

import tkinter as tk

import sqlite3

# Créer la classe principale de l'application

class AsseguimApp(tk.Tk):

# Initialiser l'application avec une fenêtre principale et un titre

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.title("Asseguim - Apprendre l'anglais de manière ludique et en équipe")

self.geometry("800x600")

self.resizable(False, False)

# Créer une variable pour stocker le mode de connexion (utilisateur ou administrateur)

self.mode = tk.StringVar()

# Créer une variable pour stocker le nom de l'utilisateur ou de l'administrateur

self.name = tk.StringVar()

# Créer une variable pour stocker l'identifiant de l'utilisateur ou de l'administrateur

self.id = tk.StringVar()

# Créer une variable pour stocker le niveau choisi par l'utilisateur (débutant, intermédiaire ou avancé)

self.level = tk.StringVar()

# Créer une variable pour stocker le score de l'utilisateur

self.score = tk.IntVar()

# Créer une variable pour stocker la question courante

self.question = tk.StringVar()

# Créer une variable pour stocker la réponse correcte à la question courante

self.answer = tk.StringVar()

# Créer une variable pour stocker la pondération de la question courante

self.weight = tk.IntVar()

# Créer une liste pour stocker les boutons des choix de réponse

self.choices = []

# Créer un dictionnaire pour stocker les divisions et leurs noms

self.divisions = {1: "Mercure", 2: "Vénus", 3: "Terre", 4: "Mars", 5: "Jupiter", 6: "Saturne", 7: "Uranus", 8: "Neptune", 9: "Pluton"}

# Créer un dictionnaire pour stocker les couleurs des divisions

self.colors = {1: "red", 2: "orange", 3: "yellow", 4: "green", 5: "blue", 6: "indigo", 7: "violet", 8: "white", 9: "black"}

# Créer une connexion à la base de données des questions

self.conn = sqlite3.connect("questions.db")

# Créer un curseur pour exécuter des requêtes SQL

self.cur = self.conn.cursor()

# Appeler la méthode qui affiche l'écran de connexion

self.show\_login\_screen()

# Définir la méthode qui affiche l'écran de connexion

def show\_login\_screen(self):

# Effacer tous les widgets de la fenêtre principale

for widget in self.winfo\_children():

widget.destroy()

# Créer un cadre pour contenir les widgets de l'écran de connexion

frame = tk.Frame(self)

frame.pack(padx=20, pady=20)

# Créer un label pour afficher le titre de l'application

title = tk.Label(frame, text="Asseguim - Apprendre l'anglais de manière ludique et en équipe", font=("Arial", 24, "bold"))

title.grid(row=0, column=0, columnspan=2, pady=10)

# Créer un label pour afficher le mode de connexion

mode\_label = tk.Label(frame, text="Mode:", font=("Arial", 16))

mode\_label.grid(row=1, column=0, sticky=tk.E)

# Créer un menu déroulant pour choisir le mode de connexion (utilisateur ou administrateur)

mode\_menu = tk.OptionMenu(frame, self.mode, "Utilisateur", "Administrateur")

mode\_menu.config(font=("Arial", 16))

mode\_menu.grid(row=1, column=1, sticky=tk.W)

# Créer un label pour afficher le nom

name\_label = tk.Label(frame, text="Nom:", font=("Arial", 16))

name\_label.grid(row=2, column=0, sticky=tk.E)

# Créer une entrée pour saisir le nom

name\_entry = tk.Entry(frame, textvariable=self.name, font=("Arial", 16))

name\_entry.grid(row=2, column=1, sticky=tk.W)

# Créer un label pour afficher l'identifiant

id\_label = tk.Label(frame, text="Identifiant:", font=("Arial", 16))

id\_label.grid(row=3, column=0, sticky=tk.E)

# Créer une entrée pour saisir l'identifiant

id\_entry = tk.Entry(frame, textvariable=self.id, font=("Arial", 16))

id\_entry.grid(row=3, column=1, sticky=tk.W)

# Créer un bouton pour valider la connexion

login\_button = tk.Button(frame, text="Se connecter", font=("Arial", 16), command=self.login)

login\_button.grid(row=4, column=0, columnspan=2, pady=10)

# Définir la méthode qui valide la connexion et affiche l'écran correspondant au mode choisi

def login(self):

# Récupérer le mode de connexion

mode = self.mode.get()

# Récupérer le nom

name = self.name.get()

# Récupérer l'identifiant

id = self.id.get()

# Vérifier que le nom et l'identifiant ne sont pas vides

if name and id:

# Si le mode est utilisateur, appeler la méthode qui affiche l'écran de choix du niveau

if mode == "Utilisateur":

self.show\_level\_screen()

# Si le mode est administrateur, appeler la méthode qui affiche l'écran de gestion des questions

elif mode == "Administrateur":

self.show\_admin\_screen()

# Sinon, afficher un message d'erreur

else:

tk.messagebox.showerror("Erreur", "Veuillez choisir un mode de connexion valide.")

# Sinon, afficher un message d'erreur

else:

tk.messagebox.showerror("Erreur", "Veuillez saisir un nom et un identifiant valides.")

# Définir la méthode qui affiche l'écran de choix du niveau pour le mode utilisateur

def show\_level\_screen(self):

# Effacer tous les widgets de la fenêtre principale

for widget in self.winfo\_children():

widget.destroy()

# Créer un cadre pour contenir les widgets de l'écran de choix du niveau

frame = tk.Frame(self)

frame.pack(padx=20, pady=20)

# Créer un label pour afficher le titre de l'application

title = tk.Label(frame, text="Asseguim - Apprendre l'anglais de manière ludique et en équipe", font=("Arial", 24, "bold"))

title.grid(row=0, column=0, columnspan=3, pady=10)

# Créer un label pour afficher le nom de l'utilisateur

name\_label = tk.Label(frame, text="Bonjour " + self.name.get(), font=("Arial", 16))

name\_label.grid(row=1, column=0, columnspan=3)

# Créer un label pour afficher le niveau choisi

level\_label = tk.Label(frame, text="Niveau:", font=("Arial", 16))

level\_label.grid(row=2, column=0, sticky=tk.E)

# Créer un menu déroulant pour choisir le niveau (débutant, intermédiaire ou avancé)

level\_menu = tk.OptionMenu(frame, self.level, "Débutant", "Intermédiaire", "Avancé")

level\_menu.config(font=("Arial", 16))

level\_menu.grid(row=2, column=1, sticky=tk.W)

# Créer un bouton pour valider le niveau et commencer le jeu

start\_button = tk.Button(frame, text="Commencer", font=("Arial", 16), command=self.start\_game)

start\_button.grid(row=2, column=2)

# Définir la méthode qui valide le niveau et commence le jeu pour le mode utilisateur

def start\_game(self):

# Récupérer le niveau choisi

level = self.level.get()

# Vérifier que le niveau n'est pas vide

if level:

# Initialiser le score à zéro

self.score.set(0)

# Appeler la méthode qui affiche l'écran du jeu

self.show\_game\_screen()

# Appeler la méthode qui génère une nouvelle question

self.new\_question()

# Appeler la méthode qui met à jour le classement des joueurs

self.update\_ranking()

# Appeler la méthode qui met à jour la division du joueur

self.update\_division()