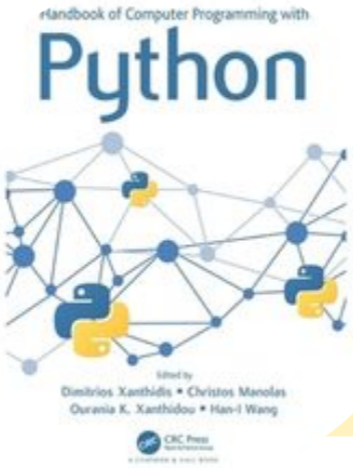
ENSAM
ÉCOLE NATIONALE SUPÉRIEURE D'ARTS ET MÉTIERS
UNIVERSITÉ HASSAN II DE CASABLANCA

# Programmation Orienté Objet

## Rapport du Mini Projet




# CourseLibrary

**application qui vous permet d'accéder à vos cours.**

## IAGI-1

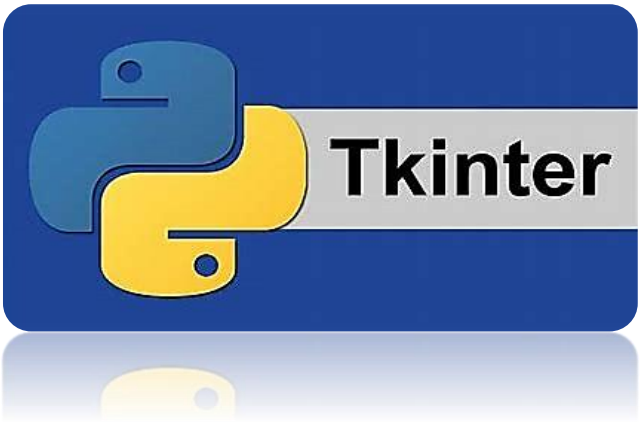
**Realiser par :**

- Handi Oumaima
- Achkhity Yassine
- Srainy Hassan

**Encadré par : Pr. MUSTAPHA HAIN**

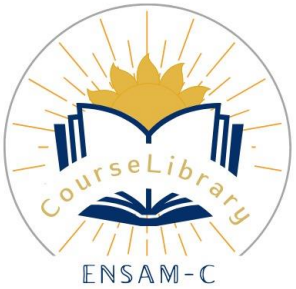
# Introduction :

Tkinter, une bibliothèque d'interfaces graphiques pour Python, joue un rôle essentiel dans le développement de projets interactifs. En tant que composant standard de la bibliothèque Python, Tkinter offre une solution puissante et conviviale pour la création d'interfaces utilisateur graphiques. Grâce à ses fonctionnalités, il permet aux développeurs de concevoir des applications avec des éléments visuels tels que des boutons, des fenêtres, des menus, et bien plus encore. L'utilité fondamentale de Tkinter réside dans sa capacité à simplifier le processus de création d'interfaces utilisateur intuitives, facilitant ainsi l'interaction entre les utilisateurs et les programmes Python. Son intégration transparente avec le langage Python en fait un choix populaire pour une variété de projets, de la conception d'applications de bureau à la création d'outils interactifs, offrant ainsi une expérience utilisateur améliorée et esthétiquement plaisante.



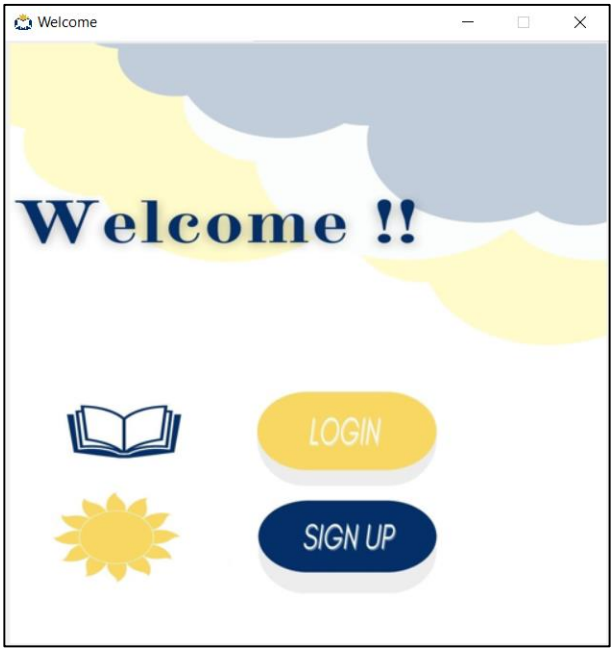

# L’objectif de l’application :

Course Library, notre application Tkinter dédiée à l'éducation, incarne un hub d'apprentissage interactif. En permettant aux utilisateurs de créer des comptes, de se connecter et d'accéder facilement à des modules de cours, Course Library offre une bibliothèque virtuelle où l'apprentissage devient accessible d'un simple clic. Les ressources éducatives, présentées sous forme de cours au format PDF, sont organisées de manière ordonnée, permettant aux utilisateurs de naviguer de manière fluide à travers les connaissances qu'ils cherchent à acquérir. Avec Courses Library, l'accès à l'éducation devient aussi simple que de tourner les pages d'une bibliothèque, mais avec la commodité d'une expérience numérique



# Description fondamentale de Course Library :

## 1. Page welcome :

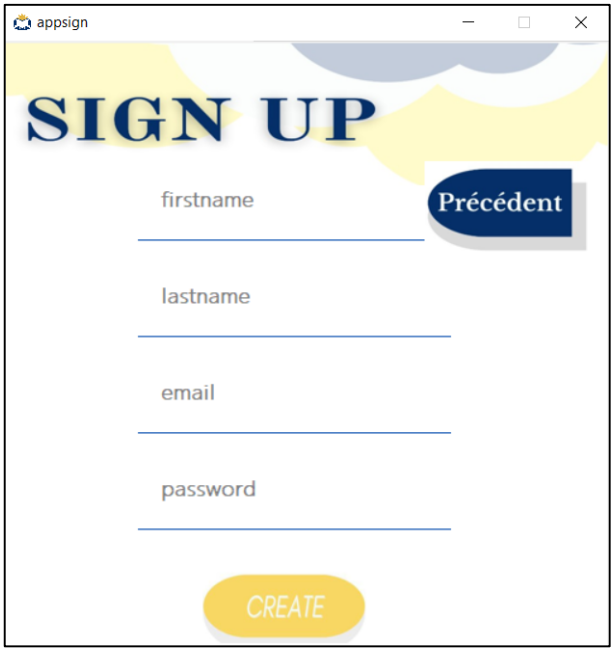



```
root = Tk()
root.title("Welcome")
root.resizable( width: False,  height: False)
icone=PhotoImage(file="pl.png")
root.iconphoto( default: False,icone)
welcome_page = resize_image('welcome.jpg')
label = Label(root, image=welcome_page)
label.pack()
frame = Frame(root, width="200", height="200", bg="#fff")
frame.place(x=190, y=280)
login_button_image = resize_image( image_path: 'loginbtn.jpg',  size: (150, 80))
login_button = Button(frame, image=login_button_image,
       activebackground="#fff",bd=0, bg="#fff", command=lambda: login(root))
login_button.place(x=20, y=10)
signUp_button_image = resize_image( image_path: 'signupbtn.jpg',  size: (150, 80))
signUp_button = Button(frame, image=signUp_button_image,
       activebackground="#fff",bd=0, bg="#fff", command=lambda: signup(root))
signUp_button.place(x=20, y=100)
container(root)
root.mainloop()
```

C'est la première page de l'application qui contient une phrase Welcome !! est deux boutons :

- **Login :** lorsque vous cliquez sur ce bouton, il vous amène vers la page Login
- **Sign Up :** lorsque vous cliquez sur ce bouton, il vous amène vers la page Sign Up

## 2. Page Sign Up :



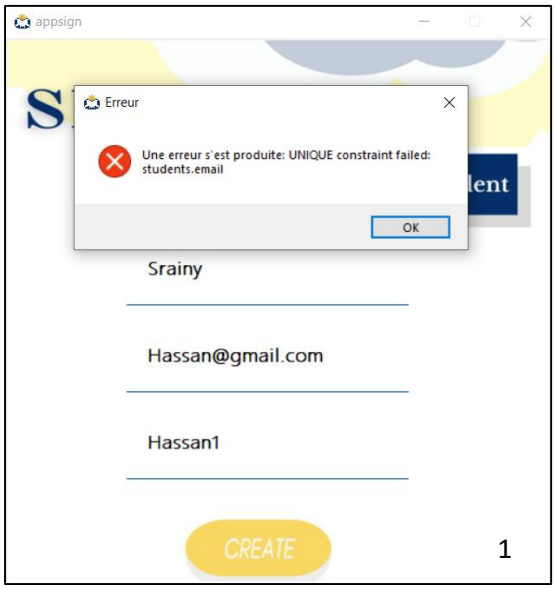

C'est la page de création de compte offre aux utilisateurs la possibilité d'entrer leurs informations personnelles telles que le prénom, le nom, l'adresse e-mail et le mot de passe. Ces données seront ensuite stockées de manière sécurisée dans une base de données SQLite3.

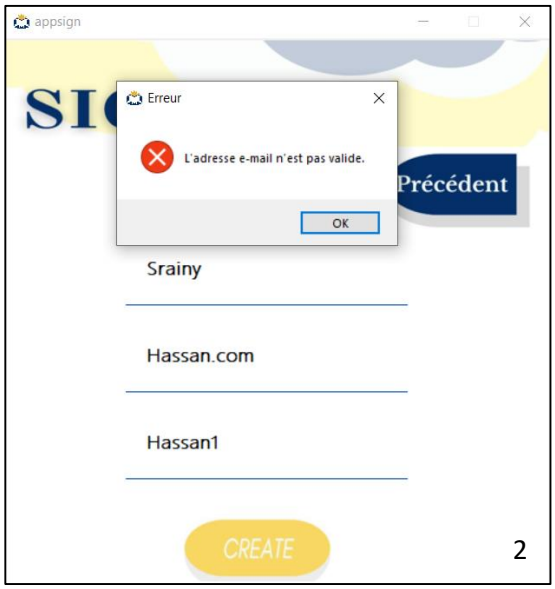
```
def create(firstname, lastname, email, password):
    if not is_valid_email(email):
        messagebox.showerror( title: 'Erreur',
         message: 'L\'adresse e-mail n\'est pas valide.')
        return False
    if not is_valid_password(password):
        messagebox.showerror( title: 'Erreur',
         message: 'Le mot de pass doit compris entre 4 et 8 caractere.')
        return False
    sql = "INSERT INTO STUDENTS (firstname, lastname, email, passwd) VALUES (?, ?, ?, ?)"
    data = (firstname.get(), lastname.get(), email.get(), password.get())
    if execute_sql(sql, data):
        messagebox.showinfo( title: 'message',  message: 'Done!')
        return True
    return False
```

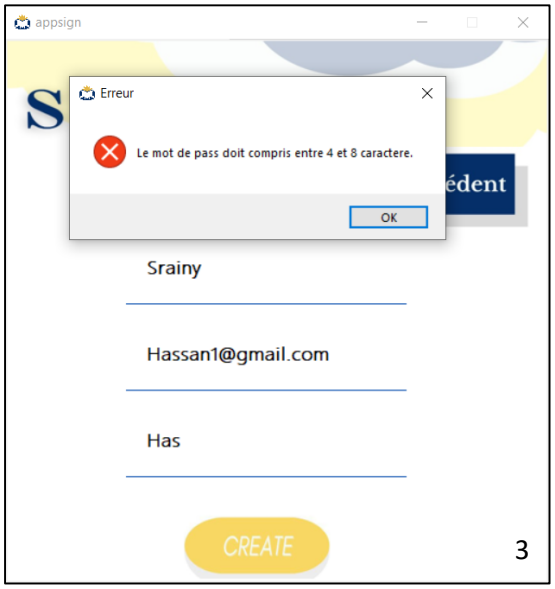
La conception de la page inclut deux boutons :

- **Create** : pour enregistrer les informations et créer le compte.
- **Précédent :** permet de revenir à la page d'accueil.

## 2.1 Gestion des Erreurs lors de la Création du Compte





**1 E-mail déjà utilisé :** L'adresse e-mail fournie par l'utilisateur est déjà enregistrée dans la base de données, ce qui signifie qu'un compte existe déjà avec cette adresse.

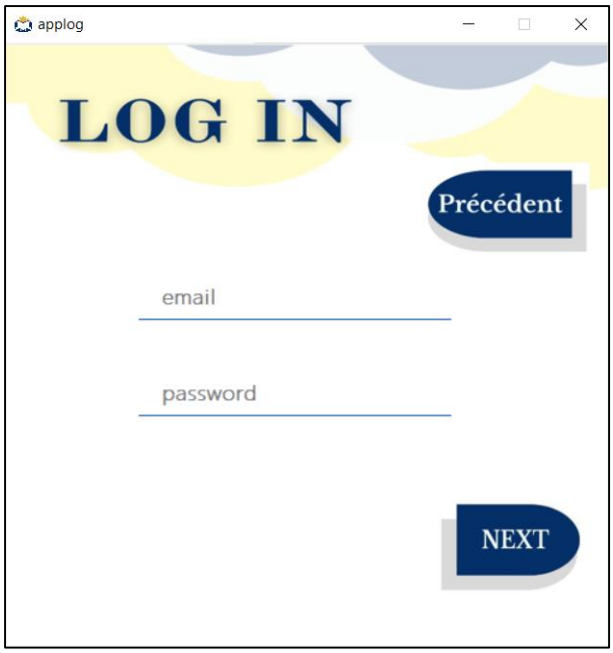
**2 E-mail invalide :** L'utilisateur a entré une adresse e-mail qui ne correspond pas au format attendu (par exemple, une adresse sans '@').

**3 Mot de passe invalide :** Le password entré par l'utilisateur ne contient pas entre 4 et 8 caractères

**4 Compte valide :** L'utilisateur a entré une adresse e-mail et un mot de passe valides, Un message s'affiche indiquant que le compte a été créé, Après la création du compte, l'application redirige automatiquement l'utilisateur vers la page des modules

```
# fonction de validation
1 usage
def is_valid_email(email_entry):
    email = email_entry.get()
    pattern = r'^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$'
    return re.match(pattern, email) is not None
1 usage
def is_valid_password(password_entry):
    password=password_entry.get()
    pattern = r'^.{4,8}$'
    return re.match(pattern, password) is not None
```

## 3. Page Login :

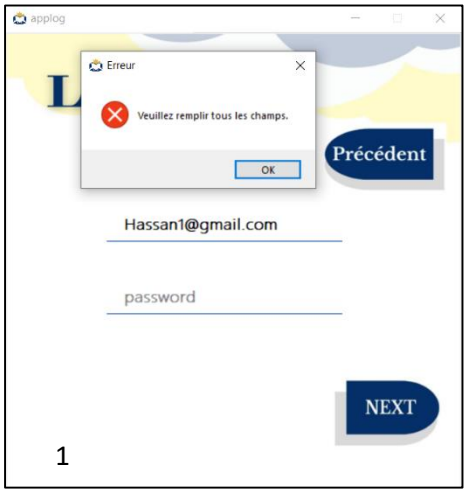



C'est la page de connexion offre aux utilisateurs un moyen sécurisé d'accéder à leur compte. Deux champs sont prévus pour saisir l'adresse e-mail et le mot de passe associés au compte.

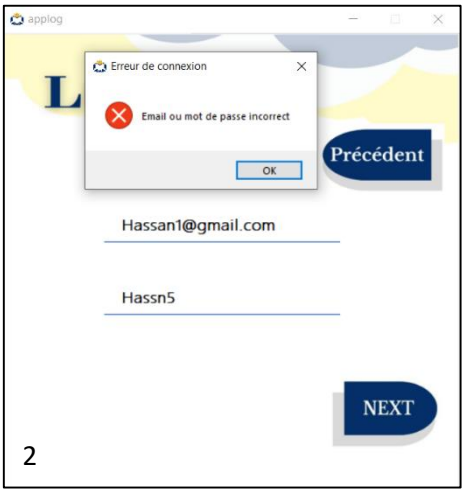
- **Next :** Les utilisateurs ont la possibilité de passer à l'étape suivante en cliquant sur ce bouton après avoir saisi leurs informations de connexion.

```
def on_submit(email_entry, password_entry):
    email = email_entry.get()
    mot_de_passe = password_entry.get()
    name=iagi.curseur.execute('SELECT firstname FROM students WHERE email=?',
                              (email,)).fetchone()
    if not check_empty_fields(email, mot_de_passe):
        return 0
    if verifier_utilisateur(email, mot_de_passe):
        messagebox.showinfo( title: "Connexion réussie"
                            , "Bienvenue, " + name[0])
        return 1
    else:
        messagebox.showerror( title: "Erreur de connexion",
                             message: "Email ou mot de passe incorrect")
        return 0
```

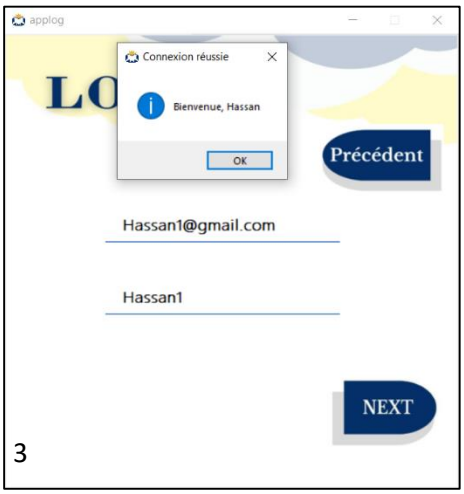
## 3.2 Gestion des Erreurs lors de la connexion :




1




2




3

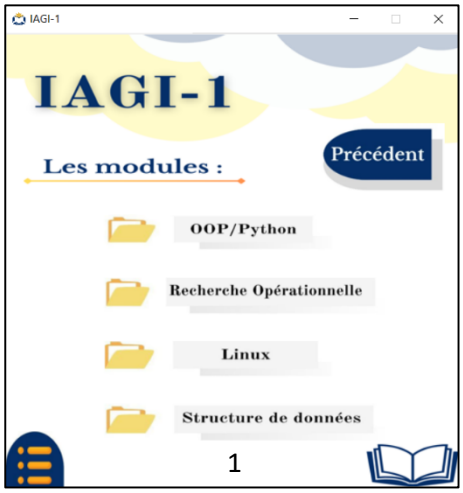
**1 Champs Non Remplis :** Lorsque l'utilisateur tente de se connecter, le système signale une erreur si l'un des champs obligatoires, tels que l'adresse e-mail ou le mot de passe, est laissé vide.

**2 Identifiants incorrects :** L'utilisateur a saisi une adresse e-mail ou un mot de passe incorrect.

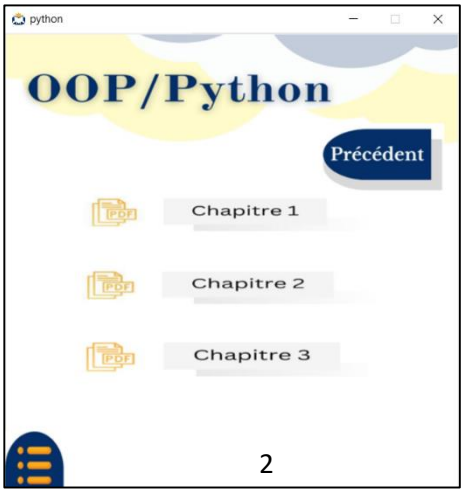
**3 Identifiants corrects :** Lorsque l'utilisateur saisit l'adresse e-mail et le mot de passe corrects, un message de bienvenue s'affiche avec le nom de l'utilisateur.et après l'application vous conduit automatiquement à la page des modules.

```
def verifier_utilisateur(email, mot_de_passe):
    iagi.curseur.execute('SELECT passwd FROM students WHERE email=?',
                         (email,))
    result = iagi.curseur.fetchone()
    if result and result[0] == mot_de_passe:
        return 1
    else:
        return 0
2 usages
def check_empty_fields(email_entry, password_entry):
    if (not email_entry or not password_entry or email_entry == "email"
            or password_entry == "password"):
        messagebox.showerror( title: "Erreur",
                              message: "Veuillez remplir tous les champs.")
        return False
    return True
```

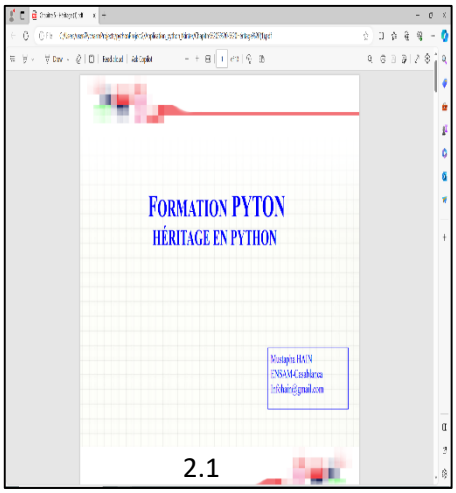
## 3.Page modules et chapitres et :



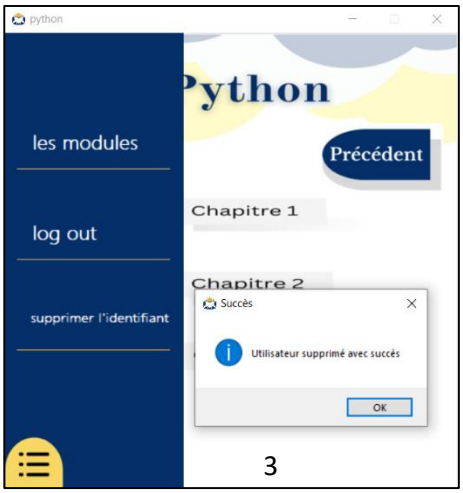
1



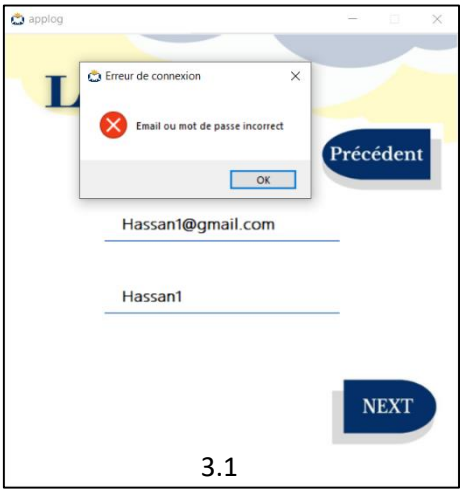
2



2.1



3



3.1

**1 Page des modules :** C'est la page qui regroupe les modules de la filière IAGI1, Cette page propose des boutons dédiés à chaque module. En cliquant sur ces boutons, vous serez dirigé(e) vers les chapitres associés à chaque module(page2).

**2 Page des chapitres :** Cette page présente les chapitres pour chaque module avec des boutons dédiés à chaque chapitre. En cliquant sur ces boutons, le chapitre s'ouvre dans un navigateur web (page2.1).

**3 La barre de navigation :** Une barre de navigation apparaît avec une animation de translation lorsqu'on clique sur le bouton du logo menu. Elle comprend trois boutons : 'Modules' pour accéder à la page des modules, 'Log Out' pour se déconnecter et revenir à la page de bienvenue, et 'Supprimer l'identifiant' pour retirer l'utilisateur de la base de données.et lorsque vous tentez de vous connecter avec le compte que vous avez supprimé, une erreur s'affiche signalant l'impossibilité de trouver l'identifiant dans la base de données (page 3.1).

```
# responsable a la suppression des utilisateurs
2 usages (1 dynamic)
def delete(email_entry):
    iagi.curseur.execute('DELETE FROM students WHERE email=?',
                         (email_entry,))
    iagi.connexion.commit()
    messagebox.showinfo( title: "Succès",
                message: "Utilisateur supprimé avec succès")
2 usages
def delete_user(email_entry):
    email_to_delete = email_entry.get()
    if email_to_delete:
        delete(email_to_delete)
        return 1
    else:
        messagebox.showwarning( title: "Avertissement",
         message: "Veuillez fournir un email pour supprimer l'utilisateur.")
        return 0
```

## Les bibliothèques utilisés dans le projet :

- **tkinter:** Permet de créer des interfaces graphiques conviviales.
- **PIL (Pillow):** Utilisé pour travailler avec des images dans l'interface graphique.
- **sqlite3:** Facilite l'interaction avec une base de données SQLite, utile pour stocker et récupérer des données.
- **subprocess:** Utilisé pour exécuter des processus externes, ce qui peut être pratique pour lancer des applications externes ou des commandes système.
- **messagebox:** Permet d'afficher des boîtes de dialogue pour communiquer des messages à l'utilisateur.
- **re:** Bibliothèque pour manipuler les expressions régulières, bien que non utilisée dans le code fourni.

## Conclusion :

En conclusion, Tkinter, en tant que bibliothèque d'interfaces graphiques pour Python, offre une solution puissante et conviviale pour le développement de projets interactifs. Son intégration transparente avec le langage Python en fait un choix populaire, permettant aux développeurs de concevoir des applications avec des interfaces utilisateur intuitives, enrichissant ainsi l'expérience des utilisateurs.

Course Library, notre application Tkinter dédiée à l'éducation, incarne un hub d'apprentissage interactif intégrant également l'utilisation de SQLite3 pour la gestion de la base de données. En permettant aux utilisateurs de créer des comptes, de se connecter et d'accéder facilement à des modules de cours, Course Library offre une bibliothèque virtuelle où l'apprentissage devient accessible d'un simple clic.

Les données des utilisateurs, ainsi que les informations sur les cours, sont stockées de manière sécurisée dans la base de données SQLite3, assurant une gestion efficace des informations. Grâce à son organisation ordonnée de ressources éducatives sous forme de cours au format PDF, Course Library simplifie l'accès à l'éducation, offrant une expérience numérique aussi pratique que feuilleter les pages d'une bibliothèque traditionnelle.

Cette application vise à rendre l'apprentissage accessible, tout en intégrant des fonctionnalités avancées de gestion de données grâce à SQLite3, offrant ainsi une plateforme moderne et esthétiquement plaisante pour l'éducation interactive.