

# APPLICATION DE GESTION UNIVERSITAIRE

---

**Projet de class  
Python**

une application de gestion universitaire qui permet d'ajouter des élèves et des filières, d'inscrire des élèves dans des filières, et d'envoyer des notifications par email.



# PRESENTATION

Ce code est un programme de gestion universitaire développé en Python utilisant la bibliothèque Tkinter pour créer une interface graphique. L'objectif principal de ce programme est de permettre la gestion des étudiants (élèves) et des filières (cours) dans un système universitaire. Voici les fonctionnalités clés et les objectifs du code :

- **Gestion des Étudiants**
- **Gestion des Filières**
- **Inscription des Étudiants**
- **Sauvegarde et chargement des Données**
- **Interface Utilisateur**
- **Notifications par Email**

The screenshot displays three windows of a university management application:

- Ajouter Élève**: A window for adding students. It contains fields for "Apogee", "Nom", and "Email", each with a corresponding input box. Below the fields are two green buttons: "Ajouter Élève" and "Afficher Élèves".
- Ajouter Filière**: A window for adding majors. It contains fields for "ID Filière" and "Nom Filière", each with a corresponding input box. Below the fields are two green buttons: "Ajouter Filière" and "Afficher Filières".
- Inscrire Élève**: A window for enrolling students. It contains dropdown menus for "Sélectionner Élève" (value: 0000) and "Sélectionner Filière" (value: F01). Below the dropdowns are two green buttons: "Inscrire Élève" and "Afficher Inscriptions".

# IMPORTATION DES BIBLIOTHÈQUES



```
1 import tkinter as tk
2 from tkinter import messagebox, ttk
3 from ttkthemes import ThemedTk
4 from PIL import Image, ImageTk
5 import csv
6 import os
7 import re
8 import smtplib
9 from email.mime.text import MIMEText
10 from email.mime.multipart import MIMEMultipart
```

#Bibliothèque standard pour créer des applications GUI

# Pour les boîtes de message et les widgets thématiques

# Pour des fenêtres Tkinter thématiques

# Pour le traitement d'images (bibliothèque Pillow)

# Pour lire et écrire des fichiers CSV

# Pour vérifier l'existence de fichiers et manipuler les chemins de fichiers

# Pour les expressions régulières (utilisées dans la validation des e-mails)

# Pour l'envoi d'e-mails via le protocole SMTP

# Pour créer du contenu d'e-mail au format texte

# Pour créer des e-mails multipart

# FONCTIONS UTILITAIRES :

`is_valid_email(email):`

**Vérifie si une adresse email est valide.**



```
1 # Email validation function
2 def is_valid_email(email):
3     pattern = r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'
4     return re.match(pattern, email) is not None
5
```

# FONCTIONS UTILITAIRES:

# load\_eleves():

**charge les données des élèves à partir du fichier csv.**



# FONCTIONS UTILITAIRES :

**save\_eleves\_to\_csv()**

**Enregistre les données des élèves dans le fichier csv.**



```
1 def save_eleves_to_csv():
2     with open(CSV_FILE, mode='w', newline='', encoding='utf-8') as file:
3         writer = csv.DictWriter(file, fieldnames=['Apogee', 'Nom', 'Email'])
4         writer.writeheader()
5         for apogee, info in eleves.items():
6             writer.writerow({
7                 'Apogee': apogee,
8                 'Nom': info['nom'],
9                 'Email': info['email']
10            })
```

# FONCTIONS UTILITAIRES :

**send\_email(to\_email, filiere\_name, student\_name)**  
**Envoie un email de confirmation d'inscription à un élève.**



```
1 def send_email(to_email, filiere_name, student_name):
2     from_email = "aymanebcontact1@gmail.com" # Replace with your email
3     from_password = "bujk tbtg btxs cqez" # Replace with your app password
4
5     subject = "Confirmation d'inscription"
6     body = f"Cher {student_name},\n\nVous avez été inscrit avec succès dans la filière {filiere_name}"
7
8     msg = MIMEMultipart()
9     msg['From'] = from_email
10    msg['To'] = to_email
11    msg['Subject'] = subject
12    msg.attach(MIMEText(body, 'plain'))
13
14    try:
15        with smtplib.SMTP('smtp.gmail.com', 587) as server:
16            server.starttls()
17            server.login(from_email, from_password)
18            server.send_message(msg)
19            print("Email envoyé avec succès.")
20    except Exception as e:
21        print(f"Échec de l'envoi de l'email : {e}")
```

**Fonctions de  
gestion des  
élèves:**

# Fonctions de gestion des élèves :

**display\_eleves()**

**Affiche la liste des élèves dans une boîte de message.**



```
1 def display_eleves():
2     if not eleves:
3         messagebox.showinfo("Élèves", "Aucun élève trouvé.")
4     else:
5         eleve_list = "\n".join([f"Apogee: {apogee}, Nom: {info['nom']}, Email: {info['email']}"]
6                               for apogee, info in eleves.items())
7     messagebox.showinfo("Élèves", eleve_list)
```

# add\_eleve()

Ajoute un nouvel élève à la liste et au fichier csv.

```
1  def add_eleve():
2      apogee = entry_eleve_apogee.get()
3      nom = entry_eleve_nom.get()
4      email = entry_eleve_email.get()
5
6      if apogee and nom and email:
7          if not is_valid_email(email):
8              messagebox.showerror("Erreur", "Veuillez entrer une adresse email valide.")
9              return
10
11     if apogee not in eleves:
12         eleves[apogee] = {
13             "nom": nom,
14             "email": email,
15             "filieres": []
16         }
17     messagebox.showinfo("Succès", f"L'élève {nom} a été ajouté avec succès!")
18     update_eleve_combobox()
19     save_eleves_to_csv()
20 else:
21     messagebox.showerror("Erreur", f"Un élève avec l'Apogee {apogee} existe déjà.")
22 else:
23     messagebox.showerror("Erreur", "Veuillez entrer l'Apogee, le nom et l'email.")
24
entry_eleve_apogee.delete(0, tk.END)
entry_eleve_nom.delete(0, tk.END)
entry_eleve_email.delete(0 , tk.END)
```

# Fonctions de gestion des élèves :

**update\_eleve\_combobox()**

**Met à jour la liste déroulante des élèves.**



```
1 # Update student combobox
2 def update_eleve_combobox():
3     combobox_enroll_eleve_apogee['values'] = list(eleves.keys())
4     if eleves:
5         combobox_enroll_eleve_apogee.set(list(eleves.keys())[0])
```

**Fonction  
d'inscription:**

# FONCTIONS D'INSCRIPTION :

**enroll\_eleve()**

**Inscrit un élève dans une filière et envoie un email de confirmation.**



```
1  def enroll_eleve():
2      apogee = combobox_enroll_eleve_apogee.get()
3      filiere_id = combobox_enroll_filiere_id.get()
4
5      if apogee and filiere_id:
6          if apogee in eleves and filiere_id in filieres:
7              if filiere_id not in eleves[apogee]["filieres"]:
8                  eleves[apogee]["filieres"].append(filiere_id)
9                  messagebox.showinfo("Succès", f"Élève inscrit avec succès dans {filieres[filiere_id]}!")
10                 send_email(eleves[apogee]["email"], filieres[filiere_id], eleves[apogee]["nom"])
11                 save_eleves_to_csv()
12             else:
13                 messagebox.showerror("Erreur", "L'élève est déjà inscrit dans cette filière.")
14         else:
15             messagebox.showerror("Erreur", "Apogee de l'élève ou ID de la filière invalide.")
16     else:
17         messagebox.showerror("Erreur", "Veuillez sélectionner à la fois l'élève et la filière.")
```

# Configuration de l'interface graphique :



Thanks for your  
attention.



# Any questions ?



Ask Google :)



# the Team



Aymane  
Bouhou



Seddik  
Boumhammdi



Abderrafea  
El Amri

---

Encadre par : Pr.Sofia Alami Kamourri