

# FONCTION D'AFFECTATION ET D'ENREGISTREMENT

---

REALISE ET PRESENTE PAR:

*AZIRI FATIMA-EZZAHRAE*

*LOUAZRI OMAR*



# Plan:



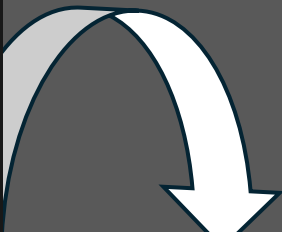
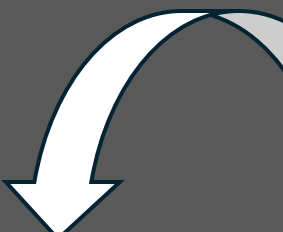
- Description de la fonction
- Exemple d'utilisation
- Présentation du code
  - Décomposition du code
  - Commentaire
- Conclusion

# Description de la fonction



# Exemples d'utilisation





```
1. Connection tant que Eleve.  
2. Connection tant que Administrateur.  
Donner l'ordre : █
```

```
1. Connection tant que Eleve.  
2. Connection tant que Administrateur.  
Donner l'ordre : 1█
```

```
Bienvenue, Donner votre ID : 32█
```

```
Bienvenue  Noir Gabriel  
1 . Afficher Mon classement  
2 . Afficher mon classement filieres  
3 . Editer mon classement filiere  
4 . Afficher la filiere affecter  
5 . Deconnection  
Donner l'ordre : █
```

```
1. Connection tant que Eleve.  
2. Connection tant que Administrateur.  
Donner l'ordre : 2█
```

```
1 . Ajouter un eleve  
2 . Editer les donnees d'un eleve  
3 . Supprimer un eleve  
4 . Afficher le classement des eleves  
5 . Afficher les Filieres  
6 . Ajouter une Filiere  
7 . Supprimer une Filiere  
8 . Affecter les eleves aux filieres  
9 . Deconnection  
Donner l'ordre : █
```

# *Présentation du code*



Le module `json` permet de lire et écrire des fichiers JSON, utilisés pour stocker des informations sur les élèves et les filières.

Le module `os` est utilisé pour effacer l'écran (`os.system('cls')`), rendant l'interface plus propre lors des changements de menu.

cette commande lit et charge les informations du fichier JSON dans la variable `data`.

```
import json
import os
```

```
# ----- INITIALISATION ----- #
```

```
with open('eleves.json', 'r') as file:
```

```
    data = json.load(file)
```

```
# Je pense que c'est mieux de mettre les filieres dans un fichier json
```

```
# ----- ADMIN ----- #
```

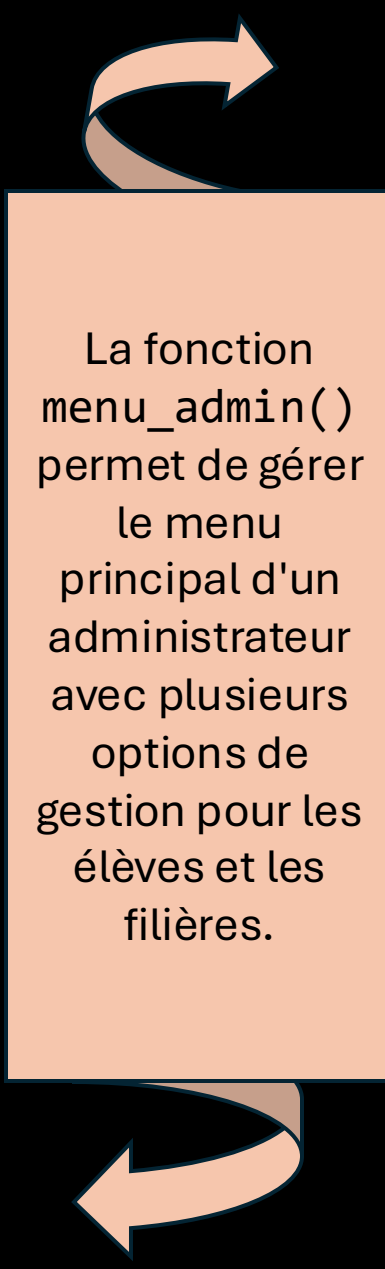
La fonction `menu_principal()` permet à l'utilisateur de choisir son type de connexion (élève ou administrateur).

Cette commande utilise le module `os` pour exécuter la commande système `cls`, qui efface l'écran de la console. Cela permet d'afficher un menu propre en début de fonction.

```
def menu_principal():  
    os.system('cls')  
    print('1. Connection tant que Eleve.')  
    print('2. Connection tant que Administrateur.')  
    n = int(input('Donner l\'ordre : '))  
    if n==1:  
        os.system('cls')  
        menu_eleve(eleve=call_eleve())  
    elif n==2:  
        os.system('cls')  
        menu_admin()  
    else:  
        os.system('cls')  
        menu_principal()
```



```
def menu_admin():
    #os.system('cls')
    print('1 . Ajouter un eleve')
    print('2 . Editer les donnees d\'un eleve')
    print('3 . Supprimer un eleve')
    print('4 . Afficher le classement des eleves')
    print('5 . Afficher les Filières')
    print('6 . Ajouter une Filiere')
    print('7 . Supprimer une Filiere')
    print('8 . Affecter les eleves aux filieres')
    print('9 . Deconnection')
    n = int(input('Donner l\'ordre : '))
    if n==1:
        os.system('cls')
        ajouter_eleve()
    elif n==2:
        os.system('cls')
        editer_eleve()
    elif n==3:
        os.system('cls')
        supprimer_eleve()
    elif n==4:
```



La fonction  
menu\_admin()  
permet de gérer  
le menu  
principal d'un  
administrateur  
avec plusieurs  
options de  
gestion pour les  
élèves et les  
filières.

```
def menu_admin():
    elif n==5:
        os.system('cls')
        afficher_filieres()
    elif n==6:
        os.system('cls')
        ajouter_filiere()
    elif n==7:
        os.system('cls')
        supprimer_filiere()
    elif n==8:
        os.system('cls')
        affecter_eleves_filiere(1)
        return_menu_principal()
    elif n == 9:
        menu_principal()
    else:
        os.system('cls')
        menu_admin()
```

*La fonction  
return\_menu\_principal()  
permet de revenir au menu  
principal d'administration  
après avoir terminé une action  
spécifique.*

```
def return_menu_principal():  
    print('-----')  
    print('1. Retourner au Menu Principal')  
    n = int(input('Donner l\'ordre : '))  
    if n==1:  
        os.system('cls')  
        menu_admin()
```

Cette fonction vérifie si un élève avec un ID donné existe dans la liste de données des élèves.

```
def check_id(id):  
    for i in range(len(data)):  
        if data[i]['id'] == id:  
            return True  
    return False
```

À chaque itération, la fonction vérifie si l'ID de l'élève actuel (`data[i]['id']`) correspond à l'ID recherché (`id`).

Cette fonction trie la liste des élèves par ordre décroissant de leur note finale, en classant les élèves ayant les notes les plus élevées en premier.

```
def classment_eleve(data):  
    #TRIER LES ELEVES PAR ORDRE DECROISSANT  
    for i in range(len(data)):  
        for j in range(i+1, len(data)):  
            if data[i]['note_final'] < data[j]['note_final']:  
                data[i], data[j] = data[j], data[i]  
    return data
```

```
def ajouter_eleve():  
    os.system('cls')  
    print("Vous choisissez d'ajouter un eleve")  
    id = int(input("Donner l'ID de l'eleve : "))  
    if(check_id(id)):  
        print("L'eleve existe deja")  
        ajouter_eleve()  
    name = input("Donner le nom complet de l'eleve : ")  
    note = float(input("Donner la note de l'eleve : "))  
    filieres = ['INDIA', 'GBM', 'IAA'] #PAR DEFALT  
    data.append({'id': id, 'NomComplet': name, 'note_final': note, 'liste_fillicre_voulu': f  
    with open('eleves.json', 'w') as file:  
        json.dump(data, file)  
    print("L'eleve a ete ajoute avec succes")
```

Cette fonction est responsable de l'ajout d'un nouvel élève dans la liste data et de la mise à jour du fichier `eleves.json`.

*Écrit (sauvegarde) la liste data dans le fichier JSON, mettant ainsi à jour le contenu du fichier avec les informations de l'élève nouvellement ajouté.*

Cette fonction est chargée d'éditer les données d'un élève, y compris son nom et sa note finale.

```
def editer_eleve( ):
    os.system('cls')
    print("Vous choisissez d'editer les donnees d'un eleve")
    id = int(input("Donner l'ID de l'eleve : "))
    if check_id(id):
        for i in range(len(data)):
            if data[i]['id'] == id:
                eleve = data[i]
                break
        print("L'eleve a editer est : ", eleve['NomCompleet'], " : ", eleve['note_final'])
        name = input("Donner le nom complet de l'eleve : ")
        note = float(input("Donner la note de l'eleve : "))
        eleve['NomCompleet'] = name
        eleve['note_final'] = note
        with open('eleves.json', 'w') as file:
            json.dump(data, file)
        print("L'eleve a ete edite avec succes")
```

```
def supprimer_eleve():  
    os.system('cls')  
    print("Vous choisissez de supprimer un eleve")  
    id = int(input("Donner l'ID de l'eleve : "))  
    if check_id(id):  
        for i in range(len(data)):  
            if data[i]['id'] == id:  
                eleve = data[i]  
                break  
        data.remove(eleve)  
        with open('eleves.json', 'w') as file:  
            json.dump(data, file)  
        print("L'eleve a ete supprime avec succes")  
    else:  
        print("L'eleve n'existe pas")  
        menu_admin()
```

Cette fonction est conçue pour supprimer un élève existant de la liste data et mettre à jour le fichier JSON qui contient les données des élèves.

Appelle la fonction menu\_admin() pour retourner au menu principal des administrateurs après une tentative de suppression.

Cette fonction affiche le classement des élèves en fonction de leurs notes finales.

```
def afficher_classement():
    os.system('cls')
    print("Le classement des eleves est : ")
    datas = classment_eleve(data)

    for i in range(len(datas)):
        print( i+1, "\t", datas[i]['NomComple'], "\t-\t", datas[i]['note_final'])
    return_menu_principal()

def afficher_filiere():
    os.system('cls')
    print("Les filieres sont : ")
    with open('filiere.json', 'r') as file:
        filieres = json.load(file)
    for i in range(len(filieres[0]['filieres'])):
        print(filieres[0]['filieres'][i]['nomFiliere'], " - Nb de places offertes : ", filie
    return_menu_principal()
```

Affiche le nom de chaque filière et le nombre de places offertes, formaté pour une meilleure lisibilité.

Affiche le rang de l'élève (i+1), le nom complet de l'élève et sa note finale, formaté avec des tabulations pour une meilleure présentation.

Cette fonction affiche la liste des filières disponibles et le nombre de places offertes pour chacune.

Cette fonction permet à un administrateur d'ajouter une nouvelle filière en demandant le nom et le nombre de places offertes.

Charge le contenu du fichier JSON dans la variable `filieres`, ce qui permet d'accéder à la liste des filières.

Ajoute un nouveau dictionnaire contenant le nom de la filière et le nombre de places offertes à la liste des filières existantes.

Écrit la liste mise à jour des filières dans le fichier JSON.

```
def ajouter_filiere():  
    os.system('cls')  
    print("Vous choisissez d'ajouter une filiere")  
    with open('filieres.json', 'r') as file:  
        filieres = json.load(file)  
    print("Voici les filieres existantes : ")  
    for i in range(len(filieres[0]['filieres'])):  
        print(filieres[0]['filieres'][i]['nomFiliere'], " - Nb de places offertes : ", filieres[0]['filieres'][i]['NbPlace'])  
    nom = input("Donner le nom de la filiere : ")  
    nb = int(input("Donner le nombre de places offertes : "))  
    filieres[0]['filieres'].append({'nomFiliere': nom, 'NbPlace': nb})  
    with open('filieres.json', 'w') as file:  
        json.dump(filieres, file)  
    print("La filiere a ete ajoute avec succes")  
    return_menu_principal()
```



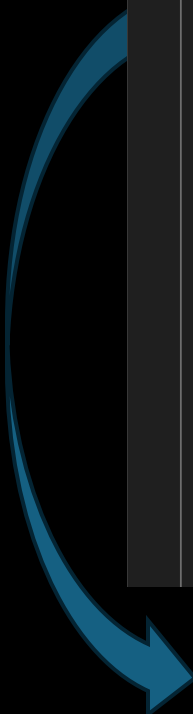
```
def supprimer_filiere():
    print("Vous choisissez de supprimer une filière")

    try:
        # Chargement des filières depuis le fichier JSON
        with open('filierees.json', 'r') as file:
            filieres = json.load(file)

        vari = filieres[0].get('filierees', [])

        # Affichage des filières existantes
        print("Voici les filières existantes :")
        for filiere in vari:
            print(f"{filiere['nomFiliere']} - Nb de places offertes : {filiere['NbPlace']}")

        # Demander le nom de la filière à supprimer
        nomf = input("Donner le nom que vous souhaitez supprimer : ")
```



Cette fonction permet à un administrateur de supprimer une filière en spécifiant son nom.

```
def supprimer_filiere():

    # Recherche et suppression de la filière
    for filiere in vari:
        if nomf == filiere['nomFiliere']:
            vari.remove(filiere)
            with open('filierees.json', 'w') as file:
                json.dump(filierees, file, indent=4)
            print("La filière a été supprimée avec succès.")
            return_menu_principal()
            return


    # Si la filière n'a pas été trouvée
    print("Cette filière n'existe pas.")

    except FileNotFoundError:
        print("Le fichier filierees.json n'existe pas.")
    except json.JSONDecodeError:
        print("Erreur de lecture du fichier JSON.")
    except Exception as e:
        print(f"Une erreur s'est produite : {e}")

    return_menu_principal()
```

```
def afficher_affectation_loop(id, nom, filiere):  
    print({'id': id, 'NomComple': nom, 'filiere': filiere})
```

Cette fonction affiche les détails d'un élève affecté  
à une filière.



```

def affecter_eleves_filiere(n=0):
    with open('filiere.json', 'r') as file:
        filieres = json.load(file)
    affectation = []
    Listes_places_restantes = {
        'INDIA': filieres[0]['filiere'][0]['NbPlace'],
        'GBM': filieres[0]['filiere'][1]['NbPlace'],
        'IAA': filieres[0]['filiere'][2]['NbPlace']
    }

    for etudiant in classment_eleve(data):
        for fil in etudiant['liste_filiere_voulu']:
            # Check if the field exists in the remaining places and has available spots
            if fil in Listes_places_restantes and Listes_places_restantes[fil] > 0:
                Listes_places_restantes[fil] -= 1 # Decrement available spots
                affectation.append({'id': etudiant['id'], 'NomComple': etudiant['NomComple'],

                if n == 1 :
                    afficher_affectation_loop( etudiant['id'], etudiant['NomComple'], fil)

                break # Move to the next student

    print("Les eleves ont ete affectes avec succes")

    print(Listes_places_restantes)

    # Write the updated affectation back to the file
    with open('affectation.json', 'w') as file:
        json.dump(affectation, file, indent=4) # Use indent for pretty printing

```

Cette fonction affecte des élèves à des filières en tenant compte des places disponibles.

```
def call_eleve():  
    os.system('cls')  
    id = int(input("Bienvenue, Donner votre ID : "))  
    if check_id(id):  
        for i in range(len(data)):  
            if data[i]['id'] == id:  
                return data[i]  
    else:  
        call_eleve()
```

*La fonction `call_eleve()` :*

- *Demande à l'élève de saisir son ID et vérifie sa validité.*
- *Si l'ID est valide, elle retourne les informations de l'élève.*
- *Si l'ID n'est pas valide, elle invite à saisir à nouveau l'ID, assurant ainsi que seul un élève avec un ID existant peut accéder à ses données.*

Fournit aux élèves un menu interactif pour accéder aux informations et options liées à leur profil.

```
def menu_eleve(eleve):
    os.system('cls')
    print('Bienvenue ', eleve['NomComple'], ' Dans votre Menu Eleve')
    print('1 . Afficher Mon classement')
    print('2 . Afficher mon classement filieres')
    print('3 . Editer mon classement filiere')
    print('4 . Afficher la filiere affecter')
    print('5 . Deconnection')
    n = int(input('Donner l\'ordre : '))
    if n==1:
        afficher_eleve_classement(eleve)
    elif n==2:
        afficher_eleve_classement_filiere(eleve)
    elif n==3:
        editer_eleve_classement_filiere(eleve)
    elif n == 4:
        afficher_filiere_affecter(eleve)
    elif n == 5:
        menu_principal()
    else:
        # Si 'n' n'est pas dans les choix, on rappelle le menu_eleve
        menu_eleve(eleve)
```

Active

Indique si un élève est affecté à une filière en vérifiant la présence de son ID dans les données d'affectation.

```
def admissible(eleve):
    with open('affectation.json', 'r') as file:
        affectation = json.load(file)
        for i in range(len(affectation)):
            if affectation[i]['id'] == eleve['id']:
                return True
    return False

def return_menu_principal_eleve(eleve):
    print('-----')
    print('1. Retourner au Menu')
    print('2. Retourner au Menu Principal')
    n = int(input('Donner l\'ordre : '))
    if n==1:
        menu_eleve(eleve)
    elif n==2:
        os.system('cls')
        menu_principal()
    else:
        return_menu_principal_eleve(eleve)
```

La 2 ème fonction affiche un sous-menu permettant à l'élève de choisir entre revenir au menu spécifique de l'élève (menu\_eleve) ou retourner au menu principal (menu\_principal).

- Fonction d'affichage de:
1. Classement d'élève
  2. Classement de filiere d'élève

La fonction combine l'affichage des informations générales de l'élève (nom et note finale), avec la liste de ses choix de filières

```
def afficher_eleve_classement(eleve):
    os.system('cls')
    print('Nom : ', eleve['NomComplet'])
    print('Note : ', eleve['note_final'])
    dataas = classment_eleve(data)
    for i in range(len(dataas)):
        if dataas[i]['id'] == eleve['id']:
            print('Votre classement est : ', i+1)
            break
    return_menu_principal_eleve(eleve)

def afficher_filieres_eleve(eleve):
    print("Votre classement des filiere est : ")
    for i in range(len(eleve['liste_filliere_voulu'])):
        print(i+1, " - ", eleve['liste_filliere_voulu'][i])
```

```
def afficher_eleve_classement_filiere(eleve):
    os.system('cls')
    print('Nom : ', eleve['NomComplet'])
    print('Note : ', eleve['note_final'])
    afficher_filieres_eleve(eleve)
    return_menu_principal_eleve(eleve)
```

Cette fonction permet à un élève de modifier son classement de filières préférées en échangeant l'ordre de deux filières.

```
def editor_eleve_classement_filiere(eleve):
    os.system('cls')
    print('Nom : ', eleve['NomComplet'])
    print('Note : ', eleve['note_final'])
    afficher_filiere_eleve(eleve)
    fil = int(input("Donner le numero de la filiere que vous souhaitez editer : "))
    replaced_fil = int(input("Donner le numero de la filiere que vous souhaitez remplacer : "))
    old_fil = eleve['liste_filliere_voulu'][fil-1]
    eleve['liste_filliere_voulu'][fil-1] = eleve['liste_filliere_voulu'][replaced_fil-1]
    eleve['liste_filliere_voulu'][replaced_fil-1] = old_fil
    with open('eleves.json', 'w') as file:
        json.dump(data, file)
        print("Votre classement a ete edite avec succes")
        affecter_eleves_filiere()
    return_menu_principal_eleve(eleve)
```

La fonction permet à un élève de voir s'il est admissible dans une filière et, le cas échéant, dans laquelle il a été affecté.

```
def afficher_filiere_affecter(eleve):
    os.system('cls')
    print('Nom : ', eleve['NomCompleet'])
    print('Note : ', eleve['note_final'])
    afficher_filieres_eleve(eleve)
    if admissible(eleve):
        print("Felicitation, Vous etes admis")
        with open('affectation.json', 'r') as file:
            affectation = json.load(file)
            for i in range(len(affectation)):
                if affectation[i]['id'] == eleve['id']:
                    print("Affectation a la filliere : ",affectation[i]['filiere'])
                    break
    else:
        print("Vous n'etes pas admissible")
    return_menu_principal_eleve(eleve)

menu_principal()
```



# { } eleves.json

```
{ } eleves.json > { } 99
1   [
2   {
3       "id": 1,
4       "NomComple": "Dupont Jean",
5       "note_final": 16.3,
6       "liste_filliere_voulu": ["GBM", "INDIA", "IAA"]
7   },
8   {
9       "id": 2,
10      "NomComple": "Anquel Jean",
11      "note_final": 13.3,
12      "liste_filliere_voulu": ["INDIA", "IAA", "GBM"]
13  },
14  {
15      "id": 3,
16      "NomComple": "Martin Sophie",
17      "note_final": 15.0,
18      "liste_filliere_voulu": ["IAA", "GBM", "INDIA"]
19  },
20  ]
```

# { } filieres.json

```
[
  {
    "filieres": [
      { "nomFiliere": "INDIA", "NbPlace": 30 },
      { "nomFiliere": "GBM", "NbPlace": 30 },
      { "nomFiliere": "IAA", "NbPlace": 30 }
    ]
  }
]
```

# { } affectation.json

```
[
  {
    "id": 1,
    "NomComple": "Noir Charles",
    "filiere": "INDIA"
  },
  {
    "id": 2,
    "NomComple": "Jourdain Celian",
    "filiere": "INDIA"
  },
  {
    "id": 3,
    "NomComple": "Clement Milys",
    "filiere": "INDIA"
  },
  {
    "id": 4,
    "NomComple": "Gosselin Camille",
    "filiere": "GBM"
  },
  {

```

***CONCLUSION***

**MERCI POUR VOTRE ATTENTION**