Exercice 1:

Écrire un programme qui permet de chercher et afficher tous les entiers « cubiques » ayant trois chiffres. Un entier naturel de trois chiffres (compris entre 100 et 999) est cubique s'il est égal à la somme des cubes de trois chiffres.

- **Exemple : 153 est cubique puisque $153 = 1^3 + 5^3 + 3^3$**

Exercice 2:

Soit une classe de 20 stagiaires. Chaque stagiaire est représenté par les informations suivantes : Nom, Prenom, Notes(math,francais,info,anglais,pc), Moyenne

Utilisez un tableau pour contenir les données des stagiaires.

On veut réaliser les traitements suivants :

- Saisir les données nécessaires
- Calculer la moyenne pour chaque stagiaire
- Trier les stagiaires par la moyenne et dans le sens décroissant en utilisant le tri par insertion
- Déterminer le classement pour chaque stagiaire
- Afficher les données de tous les stagiaires.

Exercice 3 :

Écrire un programme permettant, à l'utilisateur de **saisir les notes d'une classe**. Le programme, une fois la saisie terminée, renvoie le nombre de ces **notes supérieures à la moyenne** de la classe.

NB : Ne pas oublier des structures de contrôle pour la saisie des notes. La note doit être numérique, comprise entre 0 et 20.

Exercice 4 :

Faire un programme qui calcule le produit scalaire de deux vecteurs d'entiers U et V (de même dimension).

Exemple : (3 2 -4) * (2 -3 5) = 3 * 2 + 2 * (-3) + (-4) * 5 = -20

Exercice 5:

Écrire un programme qui affiche un triangle isocèle formé d'étoiles :

- **Exemple : Nombre d'étoiles : 9**

```
*********
*******
*****
***
*
```

Exercice 6 :

Un opérateur immobilier qui construit des groupes d'habitations désire gérer l'ensemble de ses appartements.

Chaque appartement est connu par : un numéro, une superficie et un prix de vente.

a. Saisir un certain nombre d'appartements dans la liste Appartement, chaque appartement est inséré comme un élément de type dictionnaire.
b. Trier les appartements par ordre croissant selon leurs numéros en utilisant le tri par sélection.

Il faut ensuite :

1. Afficher tous les appartements.
2. Ajouter un nouvel appartement, dont les informations sont saisies au clavier. L'ajout sera fait dans le bon endroit selon le numéro de cet appartement.
3. Supprimer un appartement dont le numéro est donné par l'utilisateur.
4. Faire une diminution de 10% du prix de vente pour les appartements ayant une superficie inférieure à 100 m^2 .
5. Sauvegarder les informations de tous les appartements dans un fichier « Appartement.txt »

- **Note : Faire un menu interactif pour accéder aux services 1, 2, 3, 4 et 5**

Exercice 7 :

Écrire un programme qui affiche la forme d'étoiles suivante :

- **Exemple : Nombre d'étoiles basique : 9**

```
    *
   **
  ***
 ****
*********
 ****
  ***
   **
    *
```

Exercice 8 :

Faire un programme qui construit le triangle de PASCAL de degré N et le mémorise dans une matrice carrée P de dimension N+1.

P = [[0],
[0, 0],
[0, 0, 0],
[0, 0, 0, 0],
[0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0]]

- **Exemple :**

  Entrez le degré du triangle de Pascal : 5

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
```

Exercice 9 :

Écrire la procédure Nchiffres qui prend comme paramètre un enter N et qui affiche le nombre de chiffres de N.

Exemple :

Introduire un nombre entier : 6457392

Le nombre 6457392 à 7 chiffres.

Écrire la fonction CubeParfait qui prend comme paramètre un entier et qui retourne 1 si cet entier est un cube parfait et 0 sinon.

Un nombre A est dit cube parfait s'il existe un entier B (B<A) tel que B3=A.

Utiliser la fonction précédente pour afficher tous les cubes parfaits inférieurs à 100.

Exercice 10 :

Écrire un programme permettant de saisir une suite d'entiers (faire le stockage dans un tableau) et de vérifier si cette suite est une suite arithmétique. Une suite (U_n) est dite arithmétique s'il existe un entier x tel que $U_{i+1} = x + U_i$; ($0<i<n$)

Exemple : la suite suivante est une suite arithmétique ($U_{i+1} = 4 + U_i$)

2 6 10 14 18 22 26 30 34 38

Exercice 11 :Tri à bulles

Exercice 12 : Tri par insertion

Exercice 13 : Tri par sélection

Exercice 14 : Étude de cas Gestion des étudiants :

Un établissement veut gérer ses étudiants inscrits et leurs notes :

Chaque étudiant est connu par son **identifiant**, son **nom**, son **prénom**, sa **filière** et sa **note finale**

Les informations de chaque étudiant sont enregistrées dans le fichier **« etudiant.txt »** comme suit :

001, ALAMI, KARIM, SRI, 12.5

002, HADI, FATIMA, CG, 15

003, RADI, SALIM, TDI, 09

Écrire un programme permettant à l'utilisateur de :

1. Saisir les nouveaux étudiants dans le fichier « etudiant.txt »
2. Supprimer l'étudiant correspondant à l'identifiant entré
3. Afficher l'ensemble des étudiants existant dans le fichier « etudiant.txt »
4. Rechercher un étudiant à l'aide de son identifiant
5. Afficher le nombre d'étudiants de chaque filière
6. Copier les étudiants réussis dans le fichier « reussi.txt » selon les critères suivants :
   - Les étudiants réussis sont ceux qui ont la note finale supérieure ou égale à 10

   Fichier « reussi.txt » :

   001, ALAMI, KARIM, SRI, 12.5

   002, HADI, FATIMA, CG, 15

- **Note : Faire un menu interactif pour accéder aux services 1, 2, 3, 4, 5 et 6.**

Exercice 15 : Étude de cas Gestion des salariés :

On souhaite écrire un programme permettant de gérer l'ensemble des salariés d'une entreprise. Chaque salarié est connu par : un **matricule**, un **nom**, un **prénom** et un **salaire**.

a- Créer une **Liste (Tableau) T** et Saisir un certain nombre de salariés dans le tableau T (chaque salarié est inséré dans le tableau comme un élément de type **dictionnaire**). Il faut ensuite :

1. Afficher tous les employés de l'entreprise.
2. Supprimer de l'entreprise un salarié dont le matricule est donné par l'utilisateur.
3. Ajouter un nouvel salarié, dont les informations sont saisies au clavier, dans le tableau T.
4. Sauvegarder les informations de tous les employés dans un fichier « Salarié. txt».

Note : Faire un menu interactif pour accéder aux services 1, 2, 3 et 4

Exercice 16 : Étude de cas Gestion des livres :

L'administrateur d'une agence de ventes de livres veut gérer les livres.

Chaque livre est connu par son numéro ISBN, son titre, son auteur, sa date de publication et son prix de ventes.

Les informations de chaque livre sont enregistrées dans le fichier **« bibliotheque.txt »** comme suit :

241600655X, Apprenez à programmer en Python, Vincent Le Goff, 24/03/2022, 320
2-212-13434-7, Apprendre à programmer avec python 3, Gérard Swinnen, 2/02/2012, 300
2-212-14088-6, Python pour les kids, Jason R. Briggs, 19/03/2015, 220

Écrire un programme permettant à l'utilisateur de :

1. Ajouter un livre à la fin du fichier "bibliotheque.txt" **(4 pts)**
2. Supprimer un livre du fichier "bibliotheque.txt" en utilisant l'ISBN du livre. **(4 pts)**
3. Afficher tous les livres présents dans le fichier "bibliotheque.txt".**(4 pts)**
4. Rechercher un livre dans le fichier "bibliotheque.txt" en utilisant son ISBN. **(4 pts)**

- **Note : Faire un menu interactif pour accéder aux services 1, 2, 3 et 4 (4 pts)**

Exercice 17 : Étude de cas Gestion des appartements :

Un opérateur immobilier qui construit des groupes d'habitations désire gérer l'ensemble de ses appartements.
Chaque appartement est connu par : un numéro, une superficie et un prix de vente.

c. Saisir un certain nombre d'appartements dans la liste Appartement, chaque appartement est inséré comme un élément de type dictionnaire.
d. Trier les appartements par ordre croissant selon leurs numéros en utilisant le tri par sélection.

Il faut ensuite :

6. Afficher tous les appartements.
7. Ajouter un nouvel appartement, dont les informations sont saisies au clavier. L'ajout sera fait dans le bon endroit selon le numéro de cet appartement.
8. Supprimer un appartement dont le numéro est donné par l'utilisateur.
9. Faire une diminution de 10% du prix de vente pour les appartements ayant une superficie inférieure à 100 m^2 .
10. Sauvegarder les informations de tous les appartements dans un fichier « Appartement.txt »

- **Note : Faire un menu interactif pour accéder aux services 1, 2, 3, 4 et 5**

Exercice 18 : L'indemnité à verser au cadre

La direction d'une entreprise désire automatiser le calcul de l'indemnité à verser aux cadres en cas de licenciement. Après d'ancienneté, dans l'entreprise, il sera alloué aux cadres licenciés une indemnité tenant compte de leur ancienneté et s'établissant comme suit :

- La moitié du salaire d'un mois par année d'ancienneté : pour la tranche d'ancienneté
- entre l ans et 10 ans.
- Au-delà de 10 ans un mois de salaire par année d'ancienneté.
- Une indemnité supplémentaire serait allouée aux cadres âgés de plus de 45 ans de :
    - 2 mois de salaire si le cadre est âgé de 46 à 49 ans.
    - 5 mois si le cadre est âgé de plus de 50 ans.

Écrire un programme python qui permet de saisir l'âge, l'ancienneté et le dernier salaire et d'afficher L’indemnité du cadre.

Correction

Exercice 1:

```
for i in range(100, 1000):
    somme = 0
    nbre = i
    while nbre > 0:
        digit = nbre % 10
        somme += digit ** 3
        nbre //= 10
    if somme == i:
        print(i)
```

Exercice 2 :

```
# Saisir les données nécessaires
stagiaires = []

while True:
    nom = input("Nom du stagiaire (ou 'exit' pour quitter): ")
    if nom == "exit":
        break
    prenom = input("Prénom du stagiaire: ")
    notemath = float(input("Note en Mathématiques : "))
    notefrancais = float(input("Note en Français : "))
    noteinfo = float(input("Note en Informatique : "))
    noteanglais = float(input("Note en Anglais : "))
    notepc = float(input("Note en PC : "))
    moyenne = (notemath + notefrancais + noteinfo + noteanglais + notepc) /
5
    stagiaires.append({"nom": nom, "prenom": prenom, "notemath": notemath,
"notefrancais": notefrancais,
                       "noteinfo": noteinfo, "noteanglais": noteanglais,
"notepc": notepc, "moyenne": moyenne})

# Trier les stagiaires par la moyenne dans le sens décroissant en utilisant
le tri par insertion
for i in range(1, len(stagiaires)):
    j = i
    while j > 0 and stagiaires[j]["moyenne"] > stagiaires[j -
1]["moyenne"]:
        stagiaires[j], stagiaires[j - 1] = stagiaires[j - 1], stagiaires[j]
        j -= 1

# Déterminer le classement pour chaque stagiaire
for i, stagiaire in enumerate(stagiaires):
    stagiaire["classement"] = i + 1

# Afficher les données de tous les stagiaires
for stagiaire in stagiaires:
    print(f"Nom: {stagiaire['nom']}")
    print(f"Prénom: {stagiaire['prenom']}")
    print(f"Note en Mathématiques: {stagiaire['notemath']}")
    print(f"Note en Français: {stagiaire['notefrancais']}")
    print(f"Note en Informatique: {stagiaire['noteinfo']}")
    print(f"Note en Anglais: {stagiaire['noteanglais']}")
    print(f"Note en PC: {stagiaire['notepc']}")
    print(f"Moyenne: {stagiaire['moyenne']}")
```

```
    print(f"Classement: {stagiaire['classement']}")
```

Exercice 3 :

```
# Initialiser une liste vide pour stocker les notes
notes = []

# Demander à l'utilisateur de saisir les notes
while True:
    note_str = input("Entrez une note (ou 'q' pour quitter) : ")
    if note_str == 'q':
        break
    try:
        note = float(note_str)
        if note < 0 or note > 20:
            print("La note doit être comprise entre 0 et 20.")
        else:
            notes.append(note)
    except ValueError:
        print("La note doit être numérique.")

# Calculer la moyenne de la classe
if len(notes) > 0:
    moyenne = sum(notes) / len(notes)
else:
    moyenne = 0
print("La moyenne de la classe est : ", moyenne)

# Compter le nombre de notes supérieures à la moyenne de la classe
nb_notes_sup = 0
for note in notes:
    if note > moyenne:
        nb_notes_sup += 1

print("Il y a", nb_notes_sup, "notes supérieures à la moyenne de la
classe.")
```

Exercice 4 :

```
def produit_scalaire(u, v):
    if len(u) != len(v):
        print("Les vecteurs doivent avoir la même dimension.")

    produit = 0
    for i in range(len(u)):
        produit += u[i] * v[i]

    return produit


# Exemple d'utilisation
u = [3, 2, -4]
v = [2, -3, 5]

resultat = produit_scalaire(u, v)
print(f"Le produit scalaire de {u} et {v} est : {resultat}")
```

Exercice 5 :

```
n=9
for i in range(n//2 + 1):
    esp=i
    et=n-2*i
    print(" " * esp,"*" * et," " * esp)
```

```
 *********
  *******
   *****
    ***
     *
```

Exercice 6 :

```
# Définition de la liste Appartement
Appartement = []


# Fonction pour trier les appartements par ordre croissant selon leurs
numéros
def tri_selection(Appartement):
    n = len(Appartement)
    for i in range(n):
        min_idx = i
        for j in range(i + 1, n):
            if Appartement[min_idx]['numero'] > Appartement[j]['numero']:
                min_idx = j
        Appartement[i], Appartement[min_idx] = Appartement[min_idx],
Appartement[i]


# Fonction pour afficher tous les appartements
def afficher_appartements(Appartement):
    for appart in Appartement:
        print(f"Numéro: {appart['numero']}, Superficie:
{appart['superficie']}, Prix de vente: {appart['prix']}")


# Fonction pour ajouter un nouvel appartement
def ajouter_appartement(Appartement):
    numero = int(input("Saisir le numéro de l'appartement : "))
    superficie = int(input("Saisir la superficie de l'appartement : "))
    prix = int(input("Saisir le prix de vente de l'appartement : "))
    nouvel_appartement = {"numero": numero, "superficie": superficie,
"prix": prix}
    Appartement.append(nouvel_appartement)
    tri_selection(Appartement)


# Fonction pour supprimer un appartement
def supprimer_appartement(Appartement):
    numero = int(input("Saisir le numéro de l'appartement à supprimer : "))
    for i, appart in enumerate(Appartement):
        if appart['numero'] == numero:
            del Appartement[i]
            Appartement.pop(i)
            Appartement.remove(appart)
            break
```

```python
# Fonction pour faire une diminution de 10% du prix de vente pour les appartements ayant une superficie inférieure à 100 m2
def diminuer_prix(Appartement):
    for appart in Appartement:
        if appart['superficie'] < 100:
            appart['prix'] = appart['prix'] * 0.9


# Fonction pour sauvegarder les informations de tous les appartements dans un fichier "Appartement.txt"
def sauvegarder_appartements(Appartement):
    with open("Appartement.txt", "w") as f:
        for appart in Appartement:

f.write(f"{appart['numero']},{appart['superficie']},{appart['prix']}\n")


# Boucle principale pour afficher le menu interactif
while True:
    print("Menu interactif :")
    print("1 - Afficher tous les appartements")
    print("2 - Ajouter un nouvel appartement")
    print("3 - Supprimer un appartement")
    print(
        "4 - Faire une diminution de 10% du prix de vente pour les appartements ayant une superficie inférieure à 100 m2")
    print("5 - Sauvegarder les informations de tous les appartements dans un fichier 'Appartement.txt'")
    print("6 - Quitter")
    choix = int(input("Saisir votre choix (1-6) : "))
    if choix == 1:
        afficher_appartements(Appartement)
    elif choix == 2:
        ajouter_appartement(Appartement)
    elif choix == 3:
        supprimer_appartement(Appartement)
    elif choix == 4:
        diminuer_prix(Appartement)
    elif choix == 5:
        sauvegarder_appartements(Appartement)
    elif choix == 6:
        break
    else:
        print("Choix invalide.")
```

Exercice 7 :

Solution 1 :

```python
n = 9
for i in range(n):
    if i < n // 2:
        espg = n // 2
        et = i + 1
        espd = n - (espg + et)
    elif i == n // 2:
```

```
        espg = 0
        espd = 0
        et = n
    else:
        espd = n // 2
        espg = i - n // 2
        et = n - (espd + espg)

    print(" " * espg, "*" * et, " " * espd)

#      *
#      **
#      ***
#      ****
# *********
#  ****
#   ***
#    **
#     *
```

Solution 2 :

```
n = 9
for i in range(n // 2):
    esp = n // 2 - 1
    et = i + 1
    print(" " * esp, "*" * et)
print("*" * n)
for i in range(n // 2):
    esp = i
    et = n // 2 - i
    print(" " * esp, "*" * et)
```

Exercice 8 :

Solution 1 :

```
N = int(input("Entrez le degré du triangle de Pascal : "))

# Initialiser P de dimension N+1 avec des zéros
#P = [[0]*(i+1) for i in range(N+1)]
P=[]
for i in range(N+1):
    P.append([0] * (i + 1))
# Remplir la première colonne de P avec des uns
for i in range(N+1):
    P[i][0] = 1

# Remplir la diagonale principale de P avec des uns
for i in range(1, N+1):
    P[i][i] = 1
# Remplir les autres cases de P en utilisant la formule de récurrence de
Pascal
for i in range(2, N+1):
    for j in range(1, i):
        P[i][j] = P[i-1][j-1] + P[i-1][j]
# Afficher P sous forme de triangle
for i in range(N+1):
    for j in range(i+1):
```

```
            print(P[i][j], end=" ")
        print()
```

Solution 2 :

```
n=int(input("Entrez le degré du triangle de Pascal : "))
p=[]
for i in range(n+1):
    el=[0]*(i+1)
    p.append(el)
for i in range(n+1):
    p[i][0]=1
    p[i][i]=1
print(p)
for i in range(2,n+1):
    for j in range(1,i):
        p[i][j]=p[i-1][j-1]+p[i-1][j]
print(p)
for el in p:
    print(el)
```

Exercice 9 :

```
def Nchiffres(N):
    # Convertir N en chaîne de caractères
    N_str = str(N)
    # Calculer le nombre de chiffres
    nb_chiffres = len(N_str)
    # Afficher le résultat
    print("Le nombre", N, "a", nb_chiffres, "chiffres.")


def CubeParfait(n):
    for i in range(1, n):
        if i ** 3 == n:
            return 1  # Le nombre est un cube parfait
    return 0  # Le nombre n'est pas un cube parfait


for i in range(1, 100):
    if CubeParfait(i) == 1:
        print(i)
```

Exercice 10 :

```
def est_suite_arithmetique(suite):
    diff = suite[1] - suite[0]  # Calcul de la différence entre les deux
premiers termes

    for i in range(2, len(suite)):
        if suite[i] - suite[i-1] != diff:
            return False

    return True
```

```
# Saisie de la suite d'entiers
suite_entiers = []
n = int(input("Entrez le nombre d'entiers dans la suite : "))

for i in range(n):
    entier = int(input("Entrez l'entier numéro {}: ".format(i+1)))
    suite_entiers.append(entier)

# Vérification si la suite est arithmétique
if est_suite_arithmetique(suite_entiers):
    print("La suite est une suite arithmétique.")
else:
    print("La suite n'est pas une suite arithmétique.")
```

Exercice 11 :Tri à bulles

```
#tri à bulles
l=[5,6,1,0,2,3,7]
n=len(l)
for i in range(n):
    for j in range(n-1):
        if l[j]>l[j+1]:
            l[j],l[j+1]=l[j+1],l[j]
    n-=1
print(l)
```

Exercice 12 : Tri par insertion

```
#tri par insertion
l=[0,1,5,6]
n=len(l)
for i in range(n):
    j=i
    while j>0 and l[j-1]>l[j]:
        l[j],l[j-1]=l[j-1],l[j]
        j=j-1
print(l)
```

Exercice 13 : Tri par sélection

```
#tri par selection
l=[5,6,1,0,2,3,7]
n=len(l)
for i in range(n):
    Imin=i
    for j in range(i+1, n):
        if l[j] < l[Imin]:
            Imin = j
    l[i], l[Imin] = l[Imin], l[i]

print(l)
```

Exercice 14 : Étude de cas Gestion des étudiants :

```
# Fonction pour ajouter un nouvel étudiant dans le fichier etudiant.txt
def ajouter_etudiant():
    with open("etudiant.txt", "a") as f:
        id = input("Saisir l'identifiant de l'étudiant: ")
        nom = input("Saisir le nom de l'étudiant: ")
        prenom = input("Saisir le prénom de l'étudiant: ")
        filiere = input("Saisir la filière de l'étudiant: ")
        note = input("Saisir la note finale de l'étudiant: ")
        f.write(f"{id}, {nom}, {prenom}, {filiere}, {note}\n")
        print("Etudiant ajouté avec succès!")

# Fonction pour supprimer un étudiant du fichier etudiant.txt
def supprimer_etudiant():
    id = input("Saisir l'identifiant de l'étudiant à supprimer: ")
    with open("etudiant.txt", "r") as f:

        for ligne in f:
            el = ligne.split(",")
            l.append({"id": el[0] , "nm": el[1], "pr": el[2], "fl": el[3],
"nt": float(el[4])})
        for lg in l:
            if lg["id" ]== id:
                l.remove(lg)
                print("Etudiant supprimé avec succès!")
                break
        else:
            print("Etudiant non trouvé!")
        with open("etudiant.txt", "w") as f:
            for lg in l:
                f.write(lg["id" ] +", " +lg["nm" ] +", " +lg["pr" ] +", "
+lg["fl" ] +", " +str(lg["nt"] ) +", " +"\n")

# Fonction pour afficher tous les étudiants du fichier etudiant.txt
def afficher_etudiants():
    with open("etudiant.txt", "r") as f:
        print("Liste des étudiants:")
        print("--------------------")
        for ligne in f:
            print(ligne.strip())

# Fonction pour rechercher un étudiant par son identifiant
def rechercher_etudiant():
    id = input("Saisir l'identifiant de l'étudiant à rechercher: ")
    with open("etudiant.txt", "r") as f:
        for ligne in f:
            if ligne.startswith(f"{id},"):
                print("Etudiant trouvé:")
                print("-----------------")
                print(ligne.strip())

        print("Aucun étudiant trouvé avec cet identifiant.")

# Fonction pour copier les étudiants réussis dans le fichier reussi.txt
def copier_etudiants_reussis():
    with open("etudiant.txt", "r") as f:
        with open("reussi.txt", "w") as f2:
            for ligne in f:
                _, _, _, _, note = ligne.split(", ")
                if float(note) >= 10:
```

```
                    f2.write(ligne)
            print("Les étudiants réussis ont été copiés dans le fichier
reussi.txt.")
# Fonction pour calculer le nombre d'étudiants pour chaque filière
def afficher_nombre_etudiants_par_filiere():
    with open('etudiant.txt', 'r') as f:
        filieres = {}
        for ligne in f:
            filiere = ligne.strip().split(', ')[3]
            if filiere in filieres:
                filieres[filiere] += 1
            else:
                filieres[filiere] = 1
        print("Nombre d'étudiants par filière:")
        for filiere, nombre in filieres.items():
            print(f"{filiere}: {nombre}")


# Programme principal
while True:
    print("\nQue voulez-vous faire?")
    print("1- Ajouter un étudiant")
    print("2- Supprimer un étudiant")
    print("3- Afficher tous les étudiants")
    print("4- Rechercher un étudiant par son identifiant")
    print("5- Copier les étudiants réussis dans le fichier reussi.txt")
    print("6- Nombre d'étudiants par filière dans le fichier reussi.txt")
    print("0- Quitter le programme")
    choix = input("Votre choix: ")

    if choix == "1":
        ajouter_etudiant()
    elif choix == "2":
        supprimer_etudiant()
    elif choix == "3":
        afficher_etudiants()
    elif choix == "4":
        rechercher_etudiant()
    elif choix == "5":
        copier_etudiants_reussis()
    elif choix == "6":
        afficher_nombre_etudiants_par_filiere()
    elif choix == "0":
        break
    else:
        print("Choix invalide")
```

Exercice 15 : Étude de cas Gestion des salariés :

```
T = []  # Initialisation de la liste des salariés

# Menu interactif
while True:
    print("1 - Afficher tous les salariés")
    print("2 - Supprimer un salarié")
    print("3 - Ajouter un nouveau salarié")
    print("4 - Sauvegarder les informations des salariés dans un fichier
texte")
    print("5 - Quitter")
    choix = input("Choix : ")
```

```
    if choix == "1":
        if len(T) == 0:
            print("La liste des salariés est vide.")
        else:
            for salarie in T:
                print("Matricule :", salarie["matricule"])
                print("Nom :", salarie["nom"])
                print("Prénom :", salarie["prenom"])
                print("Salaire :", salarie["salaire"])
                print("--------------------")

    elif choix == "2":
        matricule = input("Entrez le matricule du salarié à supprimer : ")
        for i, salarie in enumerate(T):
            if salarie["matricule"] == matricule:
                T.pop(i)
                print("Le salarié a été supprimé avec succès.")
                break
        else:
            print("Aucun salarié ne correspond au matricule donné.")

    elif choix == "3":
        matricule = input("Entrez le matricule du nouveau salarié : ")
        nom = input("Entrez le nom du nouveau salarié : ")
        prenom = input("Entrez le prénom du nouveau salarié : ")
        salaire = input("Entrez le salaire du nouveau salarié : ")
        T.append({"matricule": matricule, "nom": nom, "prenom": prenom,
"salaire": salaire})
        print("Le nouveau salarié a été ajouté avec succès.")

    elif choix == "4":
        with open("Salarié.txt", "w") as f:
            for salarie in T:

f.write(f"{salarie['matricule']},{salarie['nom']},{salarie['prenom']},{sala
rie['salaire']}\n")
        print("Les informations des salariés ont été sauvegardées dans le
fichier Salarié.txt.")

    elif choix == "5":
        print("Au revoir !")
        break

    else:
        print("Choix invalide. Veuillez réessayer.")
```

Exercice 16 : Étude de cas Gestion des livres :

```
def ajouter_livre():
    """Ajoute un livre à la fin du fichier 'bibliotheque.txt'."""
    isbn = input("Entrez le numéro ISBN du livre : ")
    titre = input("Entrez le titre du livre : ")
    auteur = input("Entrez le nom de l'auteur du livre : ")
    date = input("Entrez la date de publication du livre (au format
jj/mm/aaaa) : ")
    prix = input("Entrez le prix de vente du livre : ")
    livre = f"{isbn}, {titre}, {auteur}, {date}, {prix}\n"
    with open("bibliotheque.txt", "a") as f:
        f.write(livre)
```

```
    print("Livre ajouté avec succès.")


def supprimer_livre():
    """Supprime un livre du fichier 'bibliotheque.txt' en utilisant
l'ISBN."""
    isbn = input("Entrez le numéro ISBN du livre à supprimer : ")
    with open("bibliotheque.txt", "r") as f:
        lignes = f.readlines()
    with open("bibliotheque.txt", "w") as f:
        for ligne in lignes:
            if ligne.split(",")[0] != isbn:
                f.write(ligne)
    print("Livre supprimé avec succès.")


def afficher_livres():
    """Affiche tous les livres présents dans le fichier
'bibliotheque.txt'."""
    with open("bibliotheque.txt", "r") as f:
        for ligne in f:
            print(ligne.strip())


def rechercher_livre():
    """Recherche un livre dans le fichier 'bibliotheque.txt' en utilisant
son ISBN."""
    isbn = input("Entrez le numéro ISBN du livre à rechercher : ")
    with open("bibliotheque.txt", "r") as f:
        for ligne in f:
            if ligne.split(",")[0] == isbn:
                print(ligne.strip())
    print("Livre non trouvé.")


# Menu interactif
while True:
    print("Que voulez-vous faire ?")
    print("1. Ajouter un livre")
    print("2. Supprimer un livre")
    print("3. Afficher tous les livres")
    print("4. Rechercher un livre")
    print("5. Quitter")
    choix = input("Entrez le numéro de l'option choisie : ")
    if choix == "1":
        ajouter_livre()
    elif choix == "2":
        supprimer_livre()
    elif choix == "3":
        afficher_livres()
    elif choix == "4":
        rechercher_livre()
    elif choix == "5":
        print("Au revoir !")
        break
    else:
        print("Option invalide. Veuillez réessayer.")
```

Exercice 17 : Étude de cas Gestion des appartements :

```python
# Définition de la liste Appartement
Appartement = []


# Fonction pour trier les appartements par ordre croissant selon leurs
numéros
def tri_selection(Appartement):
    n = len(Appartement)
    for i in range(n):
        min_idx = i
        for j in range(i + 1, n):
            if Appartement[min_idx]['numero'] > Appartement[j]['numero']:
                min_idx = j
        Appartement[i], Appartement[min_idx] = Appartement[min_idx],
Appartement[i]


# Fonction pour afficher tous les appartements
def afficher_appartements(Appartement):
    for appart in Appartement:
        print(f"Numéro: {appart['numero']}, Superficie:
{appart['superficie']}, Prix de vente: {appart['prix']}")


# Fonction pour ajouter un nouvel appartement
def ajouter_appartement(Appartement):
    numero = int(input("Saisir le numéro de l'appartement : "))
    superficie = int(input("Saisir la superficie de l'appartement : "))
    prix = int(input("Saisir le prix de vente de l'appartement : "))
    nouvel_appartement = {"numero": numero, "superficie": superficie,
"prix": prix}
    Appartement.append(nouvel_appartement)
    tri_selection(Appartement)


# Fonction pour supprimer un appartement
def supprimer_appartement(Appartement):
    numero = int(input("Saisir le numéro de l'appartement à supprimer : "))
    for i, appart in enumerate(Appartement):
        if appart['numero'] == numero:
            del Appartement[i]
            Appartement.pop(i)
            Appartement.remove(appart)
            break


# Fonction pour faire une diminution de 10% du prix de vente pour les
appartements ayant une superficie inférieure à 100 m2
def diminuer_prix(Appartement):
    for appart in Appartement:
        if appart['superficie'] < 100:
            appart['prix'] = appart['prix'] * 0.9


# Fonction pour sauvegarder les informations de tous les appartements dans
un fichier "Appartement.txt"
def sauvegarder_appartements(Appartement):
    with open("Appartement.txt", "w") as f:
        for appart in Appartement:
```

```
f.write(f"{appart['numero']},{appart['superficie']},{appart['prix']}\n")


# Boucle principale pour afficher le menu interactif
while True:
    print("Menu interactif :")
    print("1 - Afficher tous les appartements")
    print("2 - Ajouter un nouvel appartement")
    print("3 - Supprimer un appartement")
    print(
        "4 - Faire une diminution de 10% du prix de vente pour les
appartements ayant une superficie inférieure à 100 m2")
    print("5 - Sauvegarder les informations de tous les appartements dans
un fichier 'Appartement.txt'")
    print("6 - Quitter")
    choix = int(input("Saisir votre choix (1-6) : "))
    if choix == 1:
        afficher_appartements(Appartement)
    elif choix == 2:
        ajouter_appartement(Appartement)
    elif choix == 3:
        supprimer_appartement(Appartement)
    elif choix == 4:
        diminuer_prix(Appartement)
    elif choix == 5:
        sauvegarder_appartements(Appartement)
    elif choix == 6:
        break
    else:
        print("Choix invalide.")
```

Exercice 18 : L'indemnité à verser au cadre

```
age = int(input("Entrez l'âge du cadre : "))
anciennete = int(input("Entrez l'ancienneté du cadre (en années) : "))
salaire = float(input("Entrez le dernier salaire du cadre : "))

# Calcul de l'indemnité de base
if anciennete >= 1 and anciennete <= 10:
    indemnite_base = (0.5 * salaire * anciennete)
else:
    indemnite_base = (salaire * anciennete)

# Calcul de l'indemnité supplémentaire pour les cadres de plus de 45 ans
if age >= 46 and age <= 49:
    indemnite_supplementaire = (2 * salaire)
elif age >= 50:
    indemnite_supplementaire = (5 * salaire)
else:
    indemnite_supplementaire = 0

# Calcul de l'indemnité totale
indemnite_totale = indemnite_base + indemnite_supplementaire

# Affichage de l'indemnité totale
print("L'indemnité à verser au cadre licencié est de :", indemnite_totale,
```

```
"euros.")



etoile = 9
n=int(etoile/2+1)
for i in range(n):
    k = etoile-2*i
    print(" " * i + "*" * k + " " * i)
```