Polynésie - mars 2021

Exercice 1 (Programmation et tris - 4 points)
Partie A : manipulation d'une liste en `Python`.

1. Donner les affichages obtenus après l'exécution du code `Python` suivant :

```
notes = [8, 7, 18, 14, 12, 9, 17, 3]
notes[3] = 16
print(len(notes))
print(notes)
```

2. Ecrire un code `Python` permettant d'afficher les éléments d'indice 2 à 4 de la liste `notes`.

Partie B : tri par insertion.
Le tri par insertion est un algorithme efficace qui s'inspire de la façon dont on peut trier une poignée de cartes. On commence avec une seule carte dans la main gauche (les autres cartes sont en tas sur la table) puis on pioche la carte suivante et on l'insère au bon endroit dans la main gauche.

1. Voici une implémentation en `Python` de cet algorithme. Recopier et compléter les lignes 6 et 7 (uniquement celles-ci).

```
1 def tri_insertion(liste):
2     """trie par insertion la liste en paramètre"""
3     for indice_courant in range(1,len(liste)):
4         element_a_inserer = liste[indice_courant]
5         i = indice_courant - 1
6         while i >= 0 and liste[i] > ...:
7             liste[...] = liste[...]
8             i = i - 1
9         liste[i + 1] = element_a_inserer
```

On a écrit dans la console les instructions suivantes :

```
notes = [8, 7, 18, 14, 12, 9, 17, 3]
tri_insertion(notes)
print(notes)
```

On a obtenu l'affichage suivant : `[3, 7, 8, 9, 12, 14, 17, 18]`
On s'interroge sur ce qui s'est passé lors de l'exécution de `tri_insertion(notes)`.

2. Donner le contenu de la liste notes après le premier passage dans la boucle `for`.
3. Donner le contenu de la liste `notes` après le troisième passage dans la boucle `for`.

Partie C : tri fusion.
L'algorithme de tri fusion suit le principe de « diviser pour régner ».

> (1) Si le tableau à trier n'a qu'un élément, il est déjà trié.
> (2) Sinon, séparer le tableau en deux parties à peu près égales.
> (3) Trier les deux parties avec l'algorithme de tri fusion.
> (4) Fusionner les deux tableaux triés en un seul tableau. source : Wikipedia

1. Cet algorithme est-il itératif ou récursif ? Justifier en une phrase.
2. Expliquer en trois lignes comment faire pour rassembler dans une main deux tas déjà triés de cartes, la carte en haut d'un tas étant la plus petite de ce même tas ; la deuxième carte d'un tas n'étant visible qu'après avoir retiré la première carte de ce tas. À la fin du procédé, les cartes en main doivent être triées

par ordre croissant.

Une fonction `fusionner` a été implémentée en `Python` en s'inspirant du procédé de la question précédente. Elle prend quatre arguments : la liste qui est en train d'être triée, l'indice où commence la sous-liste de gauche à fusionner, l'indice où termine cette sous-liste, et l'indice où se termine la sous-liste de droite.

3. Voici une implémentation de l'algorithme de tri fusion. Recopier et compléter les lignes 7, 8 et 9 (uniquement celles-ci).

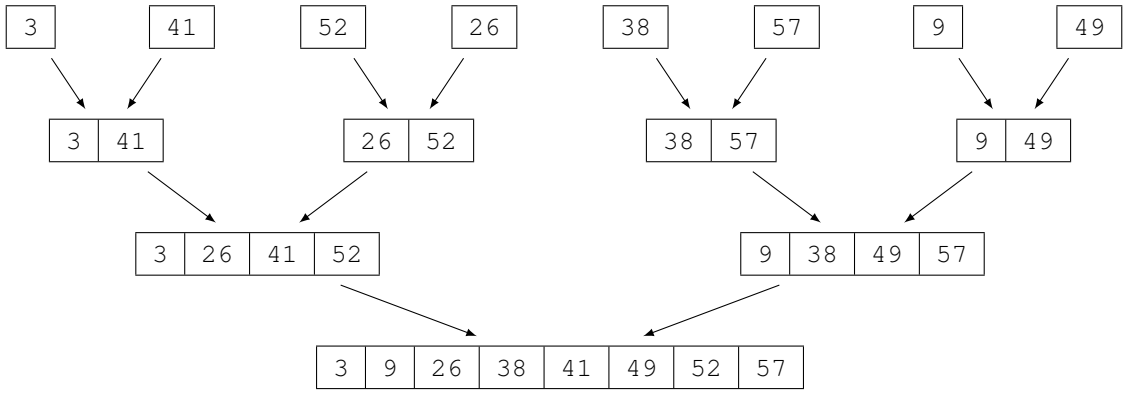
```
1 from math import floor
2
3 def tri_fusion (liste, i_debut, i_fin):
4     """trie par fusion la liste en paramètre depuis i_debut jusquà i_fin"""
5     if i_debut < i_fin:
6         i_partage = floor((i_debut + i_fin) / 2)
7         tri_fusion(liste, i_debut, ...)
8         tri_fusion(liste, ..., i_fin)
9         fusionner(liste, ..., ..., ...)
```

Remarque : la fonction `floor` renvoie la partie entière du nombre passé en paramètre.

4. Expliquer le rôle de la première ligne du code de la question 3.

Partie D : comparaison du tri par insertion et du tri fusion.

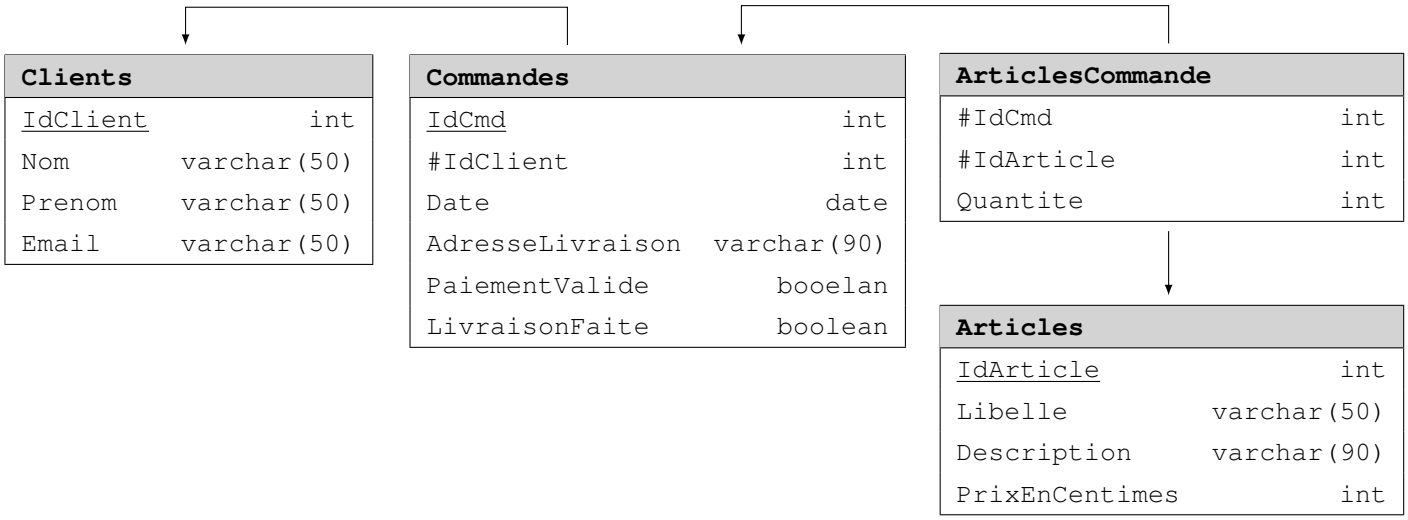
Voici une illustration des étapes d'un tri effectué sur la liste `[3, 41, 52, 26, 38, 57, 9, 49]`.



1. Quel algorithme a été utilisé : le tri par insertion ou le tri fusion ? Justifier.
2. Identifier le tri qui a une complexité, dans le pire des cas, en $O(n^2)$ et identifier le tri qui a une complexité, dans le pire des cas, en $O(n \log_2 n)$, où n représente la longueur de la liste à trier.
3. Justifier brièvement ces deux complexités.

Exercice 2 (SQL - 4 points)
Dans cet exercice, on s'intéresse à la modélisation et l'utilisation des données nécessaires aux fonctionnements de sites de vente en ligne.

Partie A : modèle relationnel.
Voici le modèle relationnel qui sera utilisé dans cet exercice :



Dans ce schéma relationnel, un attribut souligné indique qu'il s'agit d'une clé primaire. Un attibut précédé du symbole `#` indique qu'il s'agit d'une clé étrangère et la flèche associée indique la table de l'attribut référencé. Ainsi, par exemple, l'attribut `IdCmd` de la relation `ArticlesCommande` est une clé étrangère qui fait référence à l'attribut `IdCmd` de la relation `Commandes`.

1. Donner les clés primaires des relations `Clients` et `Articles`.
2. Les commandes ci-dessous ont été utilisées pour créer ces deux relations. Donner le domaine (c'est-à-dire le type) des attributs `Email` et `Quantite`.

```
CREATE TABLE Clients (
IdClient INT PRIMARY KEY,
Nom VARCHAR(50),
Prenom VARCHAR(50),
Email VARCHAR(50)
);

CREATE TABLE ArticlesCommande (
IdCmd INT,
IdArticle INT,
Quantite INT,
PRIMARY KEY (IdCmd,IdArticle)
FOREIGN KEY(IdArticle) REFERENCES Articles(IdArticle)
);
```

3. En vous inspirant des commandes ci-dessus, recopier et compléter la commande suivante qui permet de créer la relation `Commandes` en précisant sa clé étrangère.

```
CREATE TABLE Commandes (
IdCmd INT PRIMARY KEY,
IdClient INT,
Date DATE
AdresseLivraison VARCHAR(90),
PaiementValide BOOLEAN,
LivraisonFaite BOOLEAN,
FOREIGN KEY(......) REFERENCES ......
(......)
);
```

Partie B : site web.
La plateforme de vente en ligne possède un site web pour ses clients qui passent des commandes en remplissant un formulaire.

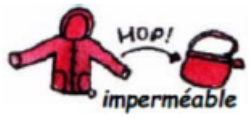
1. Expliquer pourquoi, lorsqu'un formulaire contient une quantité importante de données, il est préférable d'utiliser la méthode `POST` plutôt que `GET`.
2. Pour la validation du paiement, est-il préférable d'utiliser le protocole `HTTP` ou `HTTPS` ? Pourquoi ?
3. Expliquer l'intérêt de vérifier, avant la validation du formulaire, le format des informations saisies (par exemple qu'il n'y a pas de chiffre dans le nom ou qu'il y a bien un @ dans l'adresse de courriel).

Partie C : requêtes SQL.

1. Ecrire une requête SQL permettant de récupérer l'identifiant et le libellé de tous les articles coûtant moins de 15 euros.
2. Expliquer ce que fait la requête SQL suivante :

```
SELECT u.IdClient, u.Email, v.IdCmd, v.AdresseLivraison
FROM Clients as u JOIN Commandes as v
ON u.IdClient = v.IdClient
WHERE v.PaiementValide = False;
```

3. Ecrire une requête SQL permettant de récupérer le libellé des articles de la commande `1345`.
4. On suppose que l'attribut `IdArticle` de la table `Articles` est auto-incrémenté et ne doit donc pas être précisé lors de l'ajout d'un nouvel article. Ecrire une requête SQL permettant d'ajouter dans la base l'article suivant :



« Cet imperméable se replie en forme de pochette. »
Prix : 9,99 euros

Partie D : adaptation du modèle relationnel.

Le propriétaire du site souhaite une adaptation du modèle relationnel afin de :

- comptabiliser le stock pour chaque article ;
- pouvoir mémoriser, pour chaque client, une adresse de livraison par défaut de ses commandes.

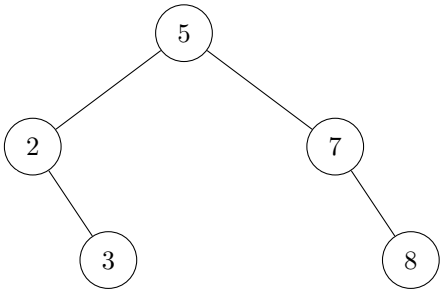
1. Préciser, pour chaque relation que vous jugez nécessaire de modifier, les attributs ajoutés ainsi que leurs domaines.
2. A l'arrivée d'une nouvelle commande d'un client, l'algorithme de mise à jour de la base de données ci-dessous est exécuté. Indiquer l'erreur présente dans cet algorithme.

```
Fonction nouvelle_commande(commande)
   Pour chaque article de la commande
      Si Quantité ⩽ Stock alors
         Stock ← Stock - 1
      Sinon
         annuler l'achat de cet article
      Fin Si
   Fin Pour
Fin Fonction
```

Exercice 3 (Arbre binaire de recherche et POO - 4 points)

Partie A : étude d'un exemple.

Considérons l'arbre binaire de recherche ci-dessous :



1. Indiquer quelle valeur a le nœud racine et quels sont les fils de ce nœud.
2. Indiquer quels sont les nœuds de la branche qui se termine par la feuille qui a pour valeur 3.
3. Dessiner l'arbre obtenu après l'ajout de la valeur 6.

Partie B : implémentation en `Python`.

Voici un extrait d'une implémentation en `Python` d'une classe modélisant un arbre binaire de recherche.

```
class ABR:
    """Implémentation dun arbre binaire de recherche (ABR)"""
    def __init__(self, valeur=None):
        self.valeur = valeur
        self.fg = None
        self.fd = None

    def estVide(self):
        return self.valeur == None

    def insererElement(self, e):
        if self.estVide():
            self.valeur = e
        else:
            if e < self.valeur:
                if self.fg:
                    self.fg.insererElement(e)
                else:
                    self.fg = ABR(e)
            if e > self.valeur:
                if self.fd:
                    self.fd.insererElement(e)
                else:
                    self.fd = ABR(e)
```

1. Expliquer le rôle de la fonction `__init__`.
2. Dans cette implémentation, expliquer ce qui se passe si on ajoute un élément déjà présent dans l'arbre.
3. Recopier et compléter les lignes de code ci-dessous permettant de créer l'arbre de la partie A.

```
arbre = ABR(......)
arbre.insererElement(2)
arbre.insererElement(......)
arbre.insererElement(7)
arbre.insererElement(......)
```

Partie C : tri par arbre binaire de recherche.

On souhaite trier un ensemble de valeurs entières distinctes grâce à un arbre binaire de recherche. Pour cela, on ajoute un à un les éléments de l'ensemble dans un arbre initialement vide. Il ne reste plus qu'à parcourir l'arbre afin de lire et de stocker dans un tableau résultat les valeurs dans l'ordre croissant.

1. Donner le nom du parcours qui permet de visiter les valeurs d'un arbre binaire de recherche dans l'ordre croissant.
2. Comparer la complexité de cette méthode de tri avec celle du tri par insertion ou du tri par sélection.

Exercice 4 (Architecture matérielle, SOC, réseaux et système d'exploitation - 4 points)
Partie A : routage dans un réseau informatique.

1. Expliquer pourquoi le protocole TCP-IP prévoit un découpage en paquets et une encapsulation des fichiers transférés d'un ordinateur à un autre via Internet.
2. On souhaite modéliser un réseau informatique par un graphe pondéré pour identifier le chemin optimal pour un paquet.
   (a) Préciser ce que représentent les sommets et les arêtes du graphe.
   (b) Préciser si le protocole RIP utilise le nombre de sauts ou le délai de réception comme poids des arêtes.

Partie B : système d'exploitation. Un système d'exploitation doit assurer la gestion des processus et des ressources.

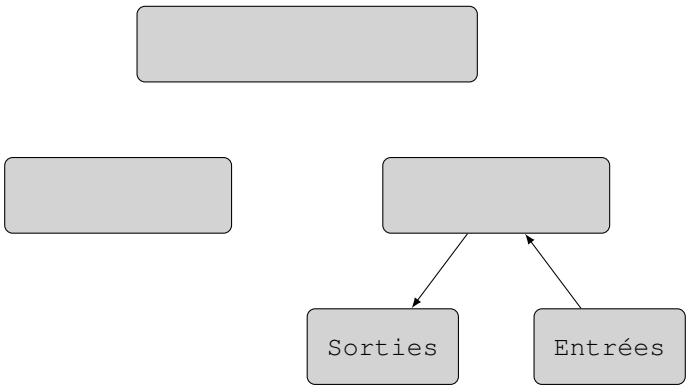
1. Dans ce contexte, expliquer et illustrer par un exemple ce qu'est une situation d'interblocage (deadlock).
2. Citer des mécanismes permettant d'éviter ces situations.

Partie C : architectures matérielles.

Architecture de Von Neumann *(source : Wikipédia)*
« L'architecture dite architecture de Von Neumann est un modèle pour un ordinateur qui utilise une structure de stockage unique pour conserver à la fois les instructions et les données demandées ou produites par le calcul. De telles machines sont aussi connues sous le nom d'ordinateur à programme enregistré. »

Elle décompose l'ordinateur en 4 éléments : l'unité de contrôle (appelé aussi unité de commande), l'unité arithmétique et logique (UAL), la mémoire et les entrées-sorties. Les deux premiers éléments sont rassemblés dans le processeur (CPU en anglais pour Control Processing Unit).

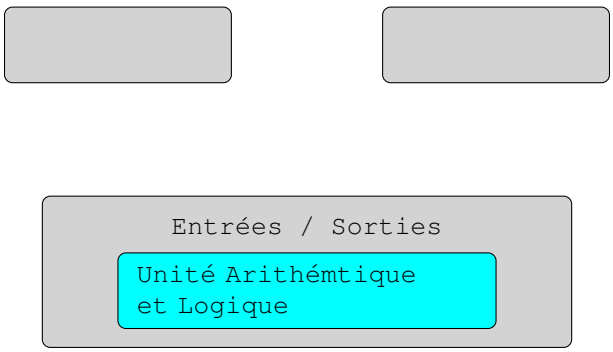
1. Recopier et compléter le schéma de cette architecture ci-dessous en faisant apparaître les communications entre les différents éléments.



2. Dans quel(s) élément(s) sont situés le « compteur de programme » (CP ou IP en anglais pour Instruction Pointer) et le « registre d'instruction » (RI ou IR en anglais pour Instruction Register). Préciser leurs rôles.

Architecture de Harvard *(source : Wikipédia)*
« L'architecture de type Harvard est une conception qui sépare physiquement la mémoire de données et la mémoire programme. L'accès à chacune des deux mémoires s'effectue via deux bus distincts. […] L'architecture Harvard est souvent utilisée dans les processeurs numériques de signal (DSP) et les microcontrôleurs. »

3. Recopier et compléter le schéma de cette architecture ci-dessous et faire apparaître les communications entre les différents éléments.



4. Expliquer ce qu'est une mémoire morte et une mémoire vive. Expliquer brièvement pourquoi, dans les microcontrôleurs, la mémoire programme est une mémoire morte.

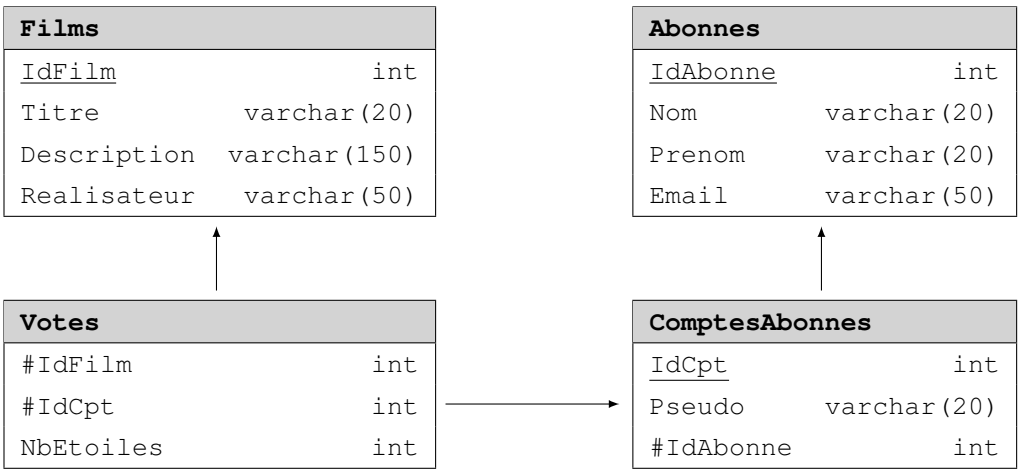
Partie D : système sur puce.

Sytème sur puce *(source : Wikipédia)*
« Un "système sur une puce", souvent désigné dans la littérature scientifique par le terme anglais "system on a chip" (d'où son abréviation SoC), est un système complet embarqué sur une seule puce (circuit intégré), pouvant comprendre de la mémoire, un ou plusieurs microprocesseurs, des périphériques d'interface, ou tout autre composant nécessaire à la réalisation de la fonction attendue. »

1. Citer un des avantages d'avoir plusieurs processeurs.
2. Expliquer pourquoi les systèmes sur puces intègrent en général des bus ayant des vitesses de transmission différentes.
3. Citer un des avantages d'un circuit imprimé de petite taille.
4. Citer un des inconvénients de cette miniaturisation.

Exercice 5 (Données en table et SQL - 4 points)
Les fournisseurs de VOD (vidéos à la demande) permettent à leurs abonnés d'avoir plusieurs comptes (afin que chaque membre de la famille puisse avoir son espace client rattaché à un compte unique pour la famille). Ils proposent également un service de votes et de conseils.

Partie A : modèle relationnel.
Voici le modèle relationnel qui sera utilisé dans cet exercice :



Dans ce schéma relationnel, un attribut souligné indique qu'il s'agit d'une clé primaire. Un attibut précédé du symbole `#` indique qu'il s'agit d'une clé étrangère et la flèche associée indique la table de l'attribut référencé. Ainsi, par exemple, l'attribut `IdCpt` de la relation `Votes` est une clé étrangère qui fait référence à l'attribut `IdCpt` de la relation `ComptesAbonnes`.

1. Donner les clés primaires des relations `Films` et `Abonnes`.
2. Le code SQL suivant a été utilisé pour créer la relation `Films`. Donner le domaine (c'est-à-dire le type) des attributs `IdFilm` et `Description`.

```
CREATE TABLE Films (
IdFilm INT AUTO_INCREMENT PRIMARY KEY,
Titre VARCHAR(20),
Description VARCHAR(150),
Realisateur VARCHAR(50)
);
```

3. Préciser la clé primaire et la clé étrangère de la relation `ComptesAbonnes`.

   L'entreprise gérant le service de VOD demande une adaptation du modèle relationnel.
4. Elle souhaite pouvoir stocker les acteurs principaux de chaque film. Préciser les modifications à apporter.
5. Elle souhaite pouvoir associer une tranche d'âge (-12 ans, 13-15 ans, 16-18 ans, +18 ans) à chacun des comptes d'un abonné et à attribuer, à chaque film, un âge minimum à avoir avant de le visionner. Préciser les modifications à apporter.

Partie B : requêtes SQL

1. Ecrire une requête SQL permettant de récupérer l'identifiant et le pseudo de tous les comptes rattachés à l'abonné 237
2. Expliquer ce que fait la requête SQL suivante :

```
SELECT AVG(NbEtoiles)
FROM Votes
WHERE IdFilm = 1542;
```

3. Expliquer ce que fait la requête SQL suivante :

```
SELECT u.IdFilm, u.Titre, v.NbEtoiles
FROM Films as u JOIN Votes as v
ON u.IdFilm = v.IdFilm
WHERE v.IdCpt = 50
ORDER BY v.NbEtoiles DESC;
```

4. Ecrire une requête SQL permettant de modifier le pseudo du compte 508 pour lui attribuer la valeur « Champion ».

Partie C : conseils de films.

Dans cette partie, nous utiliserons un module `Python` dont l'interface est fournie ci-dessous :

Module `ConsultationBaseVOD.py`	
`podiumCompte`	Prend en paramètre un identifiant `IdCpt` de compte et renvoie la liste des films les mieux notés par l'utilisateur de ce compte (au maximum 10 films) par ordre décroissant de nombre d'étoiles.
`spectateurs`	Prend en paramètre une liste `listeFilms` de films et renvoie une liste contenant les identifiants des comptes sur lesquels tous ces films ont été visionnés.
`nbEtoiles`	Prend en paramètre un identifiant `IdFilm` de film et un identifiant `IdCpt` de compte. Renvoie le nombre d'étoiles attribuées à ce film par cet utilisateur.

1. Expliquer ce que fait la fonction ci-dessous :

```
def distance(IdCpt1, IdCpt2, listeFilms):
    assert (type(listeFilms) is list) and (len(listeFilms) != 0)
    somme_ecarts = 0
    for film in listeFilms:
        somme_ecarts += abs(nbEtoiles(film, IdCpt1) - nbEtoiles(film, IdCpt2))
    return somme_ecarts / len(listeFilms)
```

2. Traduire en `Python` la fonction `conseilsFilms` suivante qui permet, à partir d'un identifiant de compte, de conseiller des films proches des goûts de l'utilisateur de ce compte.

```
Fonction conseilsFilms (IdCpt)
  Créer une liste vide nommée « conseils »
  Récupérer la liste des films les mieux notés par l'utilisateur du compte IdCpt
  Récupérer la liste des spectateurs ayant visionné ces films
  Pour chacun de ces spectateurs
    Si la distance avec l'utilisateur du compte passé en paramètre est inférieure à 10
      Ajouter à la liste « conseils » les films préférés de ce spectateur (maximum 3)
  Renvoyer la liste « conseils »
```

Remarque : Les répétitions sont autorisées dans la liste `conseils`.