BDD3 — Manipulation de tables & algorithme KNN

Données utilisées

Le tableau ci-dessous représente un extrait du jeu de données employees utilisé dans les exercices. Chaque employé est décrit par son expérience (en années), son niveau d'études (de 1 à 5), son sexe ('F' ou 'M') et son salaire (en euros).

experience	etudes	sexe	salaire
5	3	F	2400
3	3	M	2550
5	5	F	2500
3	5	M	2800
2	5	F	2300
2	3	M	2700

> *Dans les exercices suivants, la variable employes désigne ce tableau de dictionnaires, tel que défini dans donnees.py.*

Partie 1 — Manipulation de tables

Exercice 1 — Fonction salaire_moyen_condition

La fonction salaire_moyen_condition(employes, champ, valeur) doit renvoyer le salaire moyen (type float) des employés dont le champ indiqué est égal à la valeur fournie. Elle renvoie None si aucun employé ne correspond.

1. Quel est le résultat attendu des appels suivants sur le tableau employes ci-dessus ? Justifiez brièvement.

```
salaire_moyen_condition(employes, 'sexe', 'F')   →
salaire_moyen_condition(employes, 'etudes', 3)   →
salaire_moyen_condition(employes, 'etudes', 10)  →
```

2. Complétez l'implémentation de la fonction ci-dessous :

```
def salaire_moyen_condition(employes, champ, valeur):
    salaires = []
    for em in employes:
        if ____________________________________:
            ____________________________________
    if ____________________________________:
        return ____________________________________
    return ____________________________________
```

3. Écrivez une assertion permettant de vérifier que la fonction renvoie None lorsque aucun employé n'a le niveau d'études 10.

Exercice 2 — Fonction effectif_par_sexe

La fonction effectif_par_sexe(employes) renvoie un dictionnaire avec deux clés 'F' et 'M' associées respectivement au nombre de femmes et au nombre d'hommes dans le tableau.

4. Quel résultat renvoie effectif_par_sexe(employes) sur le tableau ci-dessus ?

5. Écrivez le code complet de la fonction effectif_par_sexe.

6. Proposez une assertion permettant de tester que la fonction fonctionne correctement sur le tableau employes.

Partie 2 — Analyse des écarts de salaire

Exercice 3 — Fonction calcul_ecart_sexe

On définit l'écart de salaire moyen en pourcentage comme :

$$\text{écart} = (\text{salaire moyen hommes} - \text{salaire moyen femmes}) / \text{salaire moyen hommes} \times 100$$

Voici une version incorrecte de la fonction :

```
def calcul_ecart_sexe(employes):
```

```
    moy_h = salaire_moyen_condition(employes, 'sexe', 'M')
    moy_f = salaire_moyen_condition('employes', 'sexe', 'F')  # ligne A
    return moy_h - moy_f                                       # ligne B
```

7. Identifiez et expliquez les deux erreurs présentes dans cette fonction (ligne A et ligne B).

8. Écrivez deux assertions permettant de mettre ces erreurs en évidence :
   - une assertion vérifiant que le résultat est None lorsqu'un seul sexe est présent dans la liste ;
   - une assertion vérifiant que l'écart exprimé en pourcentage est bien compris entre 0 et 100.

9. Écrivez la version corrigée de la fonction calcul_ecart_sexe.

Partie 3 — Algorithme des k plus proches voisins (KNN)

On souhaite estimer le salaire d'un nouvel employé dont on connaît l'expérience, le niveau d'études et le sexe, en s'appuyant sur les données existantes.

Exercice 4 — Calcul de la distance entre deux employés

La distance entre deux employés e1 et e2 est calculée à partir de trois critères : le sexe (converti en entier : F → 1, M → -1), l'expérience et le niveau d'études, selon la formule de la distance euclidienne.

La fonction sexe_vers_entier est déjà écrite :

```
def sexe_vers_entier(e):
    if e['sexe'] == 'F':
        return 1
    else:
        return -1
```

10. Calculez à la main la distance entre les deux employés suivants. Montrez les étapes du calcul.

```
e1 = {'experience': 3, 'etudes': 3, 'sexe': 'F'}
e2 = {'experience': 5, 'etudes': 5, 'sexe': 'M'}
```

11. Complétez le code de la fonction distance ci-dessous :

```
from math import sqrt

def distance(e1, e2):
    s = 0
    s = s + ____________________________________
    s = s + ____________________________________
    s = s + ____________________________________
    return ____________________________________
```

Exercice 5 — Tri par insertion à partir d'une clé

On dispose du code suivant de tri par insertion classique (tri d'une liste de nombres) :

```
def tri_insertion(T):
    for i in range(1, len(T)):
        val = T[i]
        j = i
        while j > 0 and T[j-1] > val:
            T[j] = T[j-1]
            j -= 1
```

```
            T[j] = val
        return T
```

On souhaite adapter ce tri pour trier une liste de tuples (distance, indice) en utilisant le premier élément du tuple comme clé de comparaison. La signature sera :

```
def tri_insertion_cle(T):
    # trie T en comparant T[i][0] (la distance)
```

12. Expliquez, en quelques lignes, le principe du tri par insertion.

13. Écrivez le code de la fonction tri_insertion_cle(T) qui trie une liste de tuples (distance, indice) par ordre croissant de distance.

Exercice 6 — Proposition de salaire par proximité

On dispose des fonctions distance, tri_insertion_cle et salaire_moyen. On veut écrire la fonction salaire_par_proximite(employes, e) qui renvoie une estimation du salaire de e en moyennant les salaires des 3 plus proches voisins parmi employes.

14. Expliquez en langage naturel comment utiliser tri_insertion_cle pour trouver les k plus proches voisins d'un employé e.

15. Complétez le code de la fonction k_plus_proches ci-dessous en utilisant tri_insertion_cle :

```
def k_plus_proches(k, employes, e):
    e_d = ____________________________________
    ____________________________________  # tri
    voisins = []
    for i in range(k):

        ____________________________________
    return voisins
```

16. En utilisant les fonctions k_plus_proches et salaire_moyen, écrivez la fonction complète salaire_par_proximite(employes, e).

17. Calculez à la main le salaire estimé pour l'employé suivant en utilisant les données du tableau en début de fiche. Montrez les distances calculées.

```
e_nouveau = {'experience': 3, 'etudes': 3, 'sexe': 'F'}
```