CORRIGÉ SUJET N°2

- ANALYSE ECART DE SALAIRES -

Le Dossier ZIP de l'épreuve pratique N°2 se situe à l'adresse suivante :
https://sujets.examens-concours.gouv.fr/delos/api/file/public/69c4efe9b11c7d6a0bc9f9f8

Ce corrigé est tiré du site suivant :
https://clem2429.github.io/Correction-Sujets-NSI-2026/

La version du fichier Python corrigé est accessible à l'adresse suivante :
https://clem2429.github.io/Correction-Sujets-NSI-2026/corrige/rle_corrige.py

- Ce sujet est le N°2 de la banque des sujets pratiques de NSI 2026;
- Ce sujet (corrigé) comporte 4 questions ;
- Chaque réponse à une question se trouve sur une page distincte.

QUESTION 1

Écrire le code de la fonction salaire_moyen_condition qui prend en paramètres un tableau d'employés dans le format décrit ci-dessus, le nom d'un des trois champs 'experience', 'etudes' ou 'sexe' ainsi qu'une valeur et qui renvoie un nombre flottant (type `float`) correspondant au salaire moyen des employés dont la valeur associée au champ est la valeur fournie.
La fonction doit renvoyer `None` s'il n'y a pas d'employés ayant la valeur recherchée pour le champ donné. Ainsi, un appel à `salaire_moyen_condition(employes, 'sexe', 'F')` permettra de renvoyer le salaire moyen des femmes employées.
Une fonction de test sur le jeu de données de test est fournie. Déterminer le salaire moyen des femmes et des hommes pour le jeu de données complet.

```
def salaire_moyen_condition(employes, champ, valeur):
    '''Renvoie le salaire moyen des employes ayant val comme
valeur associée
    au champ donné en argument.
    Si le nombre d'employés considéré est nul, cette fonction
renvoie None'''
    s = 0
    n = 0
    for employe in employes:
        if employe[champ] == valeur:
            s += employe['salaire']
            n += 1
    if n == 0:
        return None
    return s/n
```

Executer la fonction `test_salaire_moyen_condition` afin de vérifier votre fonction.
Afin de déterminer les salaires moyens des femmes et des hommes pour le jeu de données complet, on peut utiliser les appels suivants dans la console Python :

```
# On récupère les données complètes:
e = donnees_completes.employes

# Salaire moyen des femmes :
salaire_moyen_condition(e, 'sexe', 'F')
>> 2229.1961382113823
# Salaire moyen des hommes :
salaire_moyen_condition(e, 'sexe', 'M')
>> 2438.0413385826773

# Pour obtenir une moyenne des deux :
a = salaire_moyen_condition(e, 'sexe', 'F')
b = salaire_moyen_condition(e, 'sexe', 'M')
(a+b)/2
>> 2333.61873839703
```

Le salaire moyen des femmes est d'environ **2229.20** euros, tandis que celui des hommes est d'environ **2438.04** euros.
En calculant la moyenne des deux, on obtient un salaire moyen global d'environ **2333.62** euros.

QUESTION 2

Écrire en Python une fonction nommée effectif_par_sexe qui prend en paramètre un tableau non vide d'employés et qui renvoie un dictionnaire ayant deux clés 'F' et 'M' associées respectivement à l'effectif des femmes et à l'effectif des hommes employés.

Par exemple, avec le tableau employes précédent :

```
>>> effectif_par_sexe(employes)
>> {'F': 2, 'M': 3}
```

```
def effectif_par_sexe(employes):
    '''Renvoie un dictionnaire ayant deux clés 'F' et 'M'
    associée respectivement au nombre d'employées femmes et au
    nombre d'employés hommes dans les données en arguments.'''
    f = 0
    m = 0
    for employe in employes:
        if employe['sexe'] == 'F':
            f += 1
        else:
            m += 1
    return {'F': f, 'M': m}
```

QUESTION 3

écart = (salaire moyen des hommes - salaire moyen des femmes) / salaire moyen des hommes * 100

Expliquer pourquoi le code de cette fonction est incorrect et proposer quelques tests simples sous forme d'assertions qui permettent de mettre ce ou ces problèmes en évidence :
• vérifier que le résultat est None dans le cas où un seul sexe est présent ;
• vérifier qu'un écart de salaires exprimé en pourcentage soit toujours compris entre 0 et 100.

Proposer une version corrigée de la fonction ecart_salaire qui valide ces tests et renvoie le bon écart. En déduire l'écart de salaire moyen dans les données complètes.

Le code de la fonction est incorrect car il calcul mal l'écart : il calcule simplement la différence des deux. De plus, il ne gère pas les cas où un seul sexe est présent, ce qui peut entraîner une division par zéro ou un résultat incorrect.
Par ailleurs, `salaire_moyen_condition('employes', 'sexe', 'M')` ne peut marcher étant donné que `employes` est entre guillemets.

Voici quelques tests simples sous forme d'assertions pour mettre en évidence ces problèmes :

```
# Test 1 : un seul sexe présent
e = [{'sexe': 'F', 'salaire': 2000}, {'sexe': 'F', 'salaire':
2500}]
assert calcul_ecart_sexe(e) == None, "Echec du Test 1"

# Test 2 : écart de salaires exprimé en pourcentage compris
entre 0 et 100
e = [{'sexe': 'F', 'salaire': 2000}, {'sexe': 'M', 'salaire':
2500}]
assert 0 <= calcul_ecart_sexe(e) <= 100, "Echec du Test 2"

# Test 3 : écart de salaires exprimé en pourcentage correct
e = [{'sexe': 'F', 'salaire': 2000}, {'sexe': 'M', 'salaire':
2500}]
assert calcul_ecart_sexe(e) == 20, "Echec du Test 3"
```

```
# Fonction corrigée :
def calcul_ecart_sexe(employes):
    '''Renvoie l'écart de salaire en pourcentage pour les
femmes
    par rapport aux hommes'''
    # Facultatif : on vérifie que les données ne sont pas vide
:
    assert len(employes) > 0, "Le tableau d'employés ne doit
pas être vide"
    moy_h = salaire_moyen_condition(employes, 'sexe', 'M')
    moy_f = salaire_moyen_condition(employes, 'sexe', 'F')
    if moy_h is None or moy_f is None:
        return None
    return (moy_h - moy_f) / moy_h * 100
```

QUESTION 4

Afin de proposer un salaire d'embauche à un nouvel employé, le service informatique de l'entreprise a proposé d'utiliser l'algorithme des k-plus proches voisins et renvoyer la moyenne des salaires des trois employés aux caractéristiques les plus proches. La fonction `salaire_par_proximite` effectue ce calcul pour faire une proposition.

Tester et comparer les salaires proposés aux deux futurs employés suivants :

```
{'experience': 3, 'etudes': 3, 'sexe': 'F'}
{'experience': 3, 'etudes': 3, 'sexe': 'M'}
```

Identifier la source des écarts entre les deux propositions de salaire dans le programme et la corriger.

```
# On exécute dans la console :
e = donnees_completes.employes

salaire_par_proximite(e, {'experience': 3, 'etudes': 3, 'sexe':
'F'})
>> 2229.6666666666665
salaire_par_proximite(e, {'experience': 3, 'etudes': 3, 'sexe':
'M'})
>> 2406.0
```

OPTION 1 :

On remarque que la fonction `sexe_vers_entier` renvoie 1 pour les femmes et -1 pour les hommes. On distingue donc un écart de 4 entre les deux sexes.
Or, il n'y a aucune raison de différencier cela, ainsi, on peut simplement faire en sorte que la fonction `sexe_vers_entier` renvoie 1 pour les femmes et les hommes.
On peut donc modifier la fonction `sexe_vers_entier` de la manière suivante :

```
# Fonction corrigée :
def sexe_vers_entier(e):
    return 1
```

OPTION 2 :

Etant donné que la fonction corrigée `sexe_vers_entier` doit renvoyer 1, elle devient inutile. On peut donc simplement supprimer la fonction `sexe_vers_entier` et supprimer son appel par 1 dans la fonction `distance`.
On a donc la fonction modifiée `distance` suivante :

```
# Fonction corrigée :
def distance(e1, e2):
    '''Renvoie la mesure de distance entre deux personnes.'''
    s = 0
    s = s + (e1['experience'] - e2['experience'])**2
    s = s + (e1['etudes'] - e2['etudes'])**2
    return sqrt(s)
```

En conclusion, on obtient la même proposition de salaire, à savoir : `2229.6666666666665`