CORRIGÉ SUJET N°4

- CROISSANCE DES PLANTES -

Le Dossier ZIP de l'épreuve pratique N°4 se situe à l'adresse suivante :
https://sujets.examens-concours.gouv.fr/delos/api/file/public/69c4f03f3f7bb27ef4e9d9ad

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/culture_corrige_warren.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 une fonction `croissance_moyenne(plantes)` qui prend en paramètre une liste d'instances de la classe `Plante` et renvoie la moyenne des durées de croissance de l'ensemble de ces plantes (en jours). Si la liste fournie est vide, la fonction doit renvoyer `None`.
Écrire au moins deux tests pour valider le bon fonctionnement de cette fonction, dont une traitant le cas d'une liste vide.

```
def croissance_moyenne(plantes):
    if plantes == []:
        return None
    total = 0
    plante_t = len(plantes)
    for plante in plantes:
        total += plante.croissance
    return total / plante_t
```

Pour les tests possibles :

```
def test_croissance_moyenne():
    assert croissance_moyenne(plantes) == 79.0, "Echec Test 1, données plantes"
    assert croissance_moyenne([]) == None, "Echec Test 2, liste vide"
```

QUESTION 2

Écrire une fonction dictionnaire_mesure(plantes, mesures) qui prend en paramètre la liste des plantes et la liste des mesures. Elle doit renvoyer un dictionnaire où chaque clé est le nom d'une plante (présente dans la liste plantes), et chaque valeur associée est la liste des mesures concernant cette plante spécifique.
Si une plante de la liste ne possède aucune mesure associée, la liste correspondante dans le dictionnaire devra être vide. Concevoir une série de tests pertinente pour vérifier le bon comportement de cette fonction.
Exemple de retour :

```
{
    "plante1": [
        {
            "plante": "plante1",
            "jour": 1,
            "hauteur": 100,
            "temperature": 20,
            "humidite": 100,
        },
        {
            "plante": "plante1",
            "jour": 2,
            "hauteur": 101,
            "temperature": 20,
            "humidite": 100,
        },
    ],
    "plante2": [
        {
            "plante": "plante2",
            "jour": 1,
            "hauteur": 10,
            "temperature": 20,
            "humidite": 20,
        }
    ],
}
```

```python
def dictionnaire_mesure(plantes, mesures):
    dic1 = {}
    for plante in plantes:
        for mesure in mesures:
            if mesure['plante'] == plante.nom:
                if plante.nom in dic1:
                    dic1[plante.nom].append(mesure)
                else:
                    dic1[plante.nom] = [mesure]
            else:
                if not plante.nom in dic1:
                    dict1[plante.nom] = []
    return dic1
```

Afin de réaliser un test pertinent pour cette fonction, on utilisera un jeu de donné modifié et biaisé (d'autres tests sont tout de même valables), à savoir :

```
def test_dictionnaire_mesure():
    from plantes import Plante
    plantes_liste = [
        Plante("Basilic", "Ocimum basilicum", 60, 40, "plein
soleil"),
        Plante("Tomate", "Solanum lycopersicum", 80, 100,
"plein soleil"),
        Plante("Iris", "Iris", 80, 50, "plein soleil")
    ]
    mesures = [
        {'jour': 1, 'plante': 'Basilic', 'hauteur': 0.85,
'temperature': 29.3, 'humidite': 50.89},
        {'jour': 1, 'plante': 'Tomate', 'hauteur': 1.27,
'temperature': 21.51, 'humidite': 47.19}
    ]
    assert dictionnaire_mesure(plantes_liste, mesures) ==
{"Basilic": [{"plante": "Basilic","jour": 1,"hauteur":
0.85,"temperature": 29.3,"humidite": 50.89,}],"Tomate":
[{"plante": "Tomate","jour": 1,"hauteur": 1.27,"temperature":
21.51,"humidite": 47.19,}],"Iris": []}
```

QUESTION 3

Exécuter le test de la fonction `test_purger` et analyser son code pour identifier la source de cette erreur logique.

On exécute dans la console :

```
test_purger()
>>  Résultat après la purge :
    Jour 2 : 19.0°C
    Jour 3 : 22.0°C
    Jour 5 : 29.0°C
```

On remarque que l'erreur provient du .remove() qui est utilisé afin de supprimer une mesure. Le seul problème est que celui-ci modifie la liste et donc, son indexation, ce qui fait que les indices suivants ne correspondent plus à la même mesure. Le résultat est donc faussé.

QUESTION 4

Proposer une version corrigée de la fonction répondant parfaitement à l'objectif.

```
# Fonction corrigée :
def purger_mesures_extremes(liste_mesures):
    """
    Supprime de la liste toutes les mesures dont la température
    n'est pas comprise entre 20 et 25°C inclus.
    """
    copie = liste_mesures.copy()
    for mesure in copie:
        if not (20 < mesure['temperature'] < 25):
            liste_mesures.remove(mesure)
    return liste_mesures
```