BACCALAUREAT

SESSION 2022

Épreuve de l'enseignement de spécialité

NUMERIQUE et SCIENCES INFORMATIQUES

Partie pratique

Classe Terminale de la voie générale

Sujet n°40

DUREE DE L'EPREUVE : 1 heure

Le sujet comporte 3 pages numérotées de 1 / 3 à 3 / 3
Dès que le sujet vous est remis, assurez-vous qu'il est complet.

Le candidat doit traiter les 2 exercices.

EXERCICE 1 (4 points)

Écrire une fonction `recherche` qui prend en paramètres `elt` un nombre et `tab` un tableau de nombres, et qui renvoie le tableau des indices de `elt` dans `tab` si `elt` est dans `tab` et le tableau vide `[]` sinon.

Exemples :

```
>>> recherche(3, [3, 2, 1, 3, 2, 1])
[0, 3]
>>> recherche(4, [1, 2, 3])
[]
```

EXERCICE 2 (4 points)

Un professeur de NSI décide de gérer les résultats de sa classe sous la forme d'un dictionnaire :

- les clefs sont les noms des élèves ;
- les valeurs sont des dictionnaires dont les clefs sont les types d'épreuves et les valeurs sont les notes obtenues associées à leurs coefficients.

Avec :

```
resultats = {'Dupont':{'DS1' : [15.5, 4],
                       'DM1' : [14.5, 1],
                       'DS2' : [13, 4],
                       'PROJET1' : [16, 3],
                       'DS3' : [14, 4]},
             'Durand':{'DS1' : [6 , 4],
                       'DM1' : [14.5, 1],
                       'DS2' : [8, 4],
                       'PROJET1' : [9, 3],
                       'IE1' : [7, 2],
                       'DS3' : [8, 4],
                       'DS4' :[15, 4]}}
```

L'élève dont le nom est Durand a ainsi obtenu au DS2 la note de 8 avec un coefficient 4.

Le professeur crée une fonction `moyenne` qui prend en paramètre le nom d'un de ces élèves et lui renvoie sa moyenne arrondie au dixième.

Compléter le code du professeur ci-dessous :

```
def moyenne(nom):
    if nom in ...:
        notes = resultats[nom]
        total_points = ...
        total_coefficients = ...
        for ...  in notes.values():
            note , coefficient = valeurs
            total_points = total_points + ... * coefficient
            total_coefficients = ... + coefficient
        return round( ... / total_coefficients , 1 )
    else:
        return -1
```

```
"""
Author: Pascal Padilla
Source: correction de l'exercice 1 du sujet 40 des épreuves pratiques NSI 2022

Remarques:
    * bien lire l'énoncé : on veut un tableau d'indices
"""



def recherche(elt, tab: list) -> list:
    """ Renvoie le tableau des indices de elt dans tab

    Exemples et tests:
    >>> recherche(3, [3, 2, 1, 3, 2, 1])
    [0, 3]
    >>> recherche(4, [1, 2, 3])
    []
    """
    # BOUCLE: parcour de tout le tableau à la recherche de elt
    #-> invariant: `indices` est le tableau de tous les indices de elt
    #           dans la zone parcourue jusqu'à présent : tab[0 .. i-1]
    indices = []
    # -> condition d'arrêt: après la boucle i <- dernier indice de tab
    for i in range(len(tab)):
        # maintient de l'invariant
        # cas où l'élément courant correspond à `elt`
        if tab[i] == elt:
            # ajout de l'indice courant dans `indices`
            indices.append(i)

    # fin BOUCLE: tout le tableau est parcouru + invariant
    # => indices est correct
    return indices


# Vérification avec des assertions
assert  recherche(3, [3, 2, 1, 3, 2, 1]) == [0, 3]
assert recherche(4, [1, 2, 3]) == []

# Vérifications avec des affichages
print(recherche(3, [3, 2, 1, 3, 2, 1]))     # [0, 3]
print(recherche(4, [1, 2, 3]))              # []

# Vérification avec doctest et testmod
from doctest import testmod
testmod()
```

```
"""
Author: Pascal Padilla
Source: correction de l'exercice 2 du sujet 40 des épreuves pratiques NSI 2022

Remarques:
    * bien comprendre le parcours de dictionnaire `for valeurs in dico.values()`
    À chaque tour de boucle, `valeurs` vaut la valeur de la clé courante
    Ce parcours ne s'intéresse donc pas à la clé,
    mais parcourt le dictionnaire quand même.
"""



resultats = {'Dupont':{'DS1' : [15.5, 4],
                       'DM1' : [14.5, 1],
                       'DS2' : [13, 4],
                       'PROJET1' : [16, 3],
                       'DS3' : [14, 4]},
             'Durand':{'DS1' : [6 , 4],
                       'DM1' : [14.5, 1],
                       'DS2' : [8, 4],
                       'PROJET1' : [9, 3],
                       'IE1' : [7, 2],
                       'DS3' : [8, 4],
                       'DS4' :[15, 4]}}


def moyenne(nom):
    # cas de la clé nom qui est présente dans le dictionnaire
    # c'est-à-dire que le prof a bien l'élève dans ses résultats
    if nom in resultats:
        # création d'un dictionnaire contenant
        # toutes les évaluations de l'élève `nom`
        notes = resultats[nom]

        # BOUCLE: parcours de toutes les notes pour
        #         calculer le total des points et des coefficients
        # -> invariant: total_point est le cumul de tous les (points × coef)
        #               de la zone parcourue jusqu'à présent
        total_points = 0
        # -> invariant: total_coefficients est le cumul des coef
        #               de la zone parcourue jusqu'à présent
        total_coefficients = 0
        # -> condition d'arrêt: après la dernière évaluation
        for valeurs in notes.values():
            # comme valeurs est une évaluation, c'est un tableau
            # de deux élèments: (1) la note et (2) le coefficient
            # la ligne suivante est une pythonnerie qui équivaut à :
            #   note = valeurs[0]
            #   coefficient = valeurs[1]
            note , coefficient = valeurs

            # maintient des deux invariants
            total_points = total_points + note * coefficient
            total_coefficients = total_coefficients + coefficient

        # fin BOUCLE: toutes les évaluations sont parcourues

        # formule de la moyenne coefficientée
        return round( total_points / total_coefficients , 1 )

    # cas d'un élève qui n'a aucune évaluation
    # (et donc n'est pas dans le dictionnaire)
    else:
        return -1


# Vérification avec des assertions
assert moyenne('Dupont') == 14.5
assert moyenne('Dupond') == -1

# Vérification avec des affichages
print(moyenne('Dupont'))    # 14.5
```

```
print(moyenne('Dupond'))     # -1
```