IV - TP 1 Tableaux 2D

1 On considère `tab = [[9, 23, 4, 9], [22, 16, 3, 0], [5, 6, 1, 8]]`.

1. Que vaut `tab[2]` ?
2. Que vaut `tab[1][2]` ?
3. Que vaut `len(tab)` ?
4. Que vaut `len(tab[0])` ?
5. Un des éléments de `tab` vaut 4.

   Donner une instruction pour le remplacer par 14.
6. Expliquer précisément ce que fait la fonction suivante :

Code python

```
def mystere(tab):
    """ [[int]] -> None """
    m, n = len(tab), len(tab[0])
    for i in range(m):
        for j in range(n):
            tab[i][j] = 2*tab[i][j]
```

2 1. Écrire une fonction `nombre_occurrences` qui prend en argument un tableau 2D d'entiers `tab` et un entier `e` et qui renvoie le nombre de fois que l'élément `e` est présent dans `tab`.

Code python

```
def nombre_occurrences(tab, e):
    """ [[int]], int -> int
    Renvoie le nombre d'occurrences de e dans tab """
    pass
```

Code python

```
tab = [[3, 3, 3], [3, 2, 1], [0, 1, 2], [2, 1, 3]]
for e in range(5):
    print(nombre_occurrences(tab, e), end = ' ')
```

Résultat

```
1 3 3 5 0
```

2. Écrire une fonction `nombre_occurrences_col` qui prend en argument un tableau 2D d'entiers `tab`, un entier `j` correspondant à un indice de colonne et un entier `e` et qui renvoie le nombre de fois que l'élément `e` est présent dans la colonne d'indice `j` de `tab`.

Code python

```
def nombre_occurrences_col(tab, j, e):
    """ [[int]], int, int -> int
    Renvoie le nombre d'occurrences de e dans la colonne j """
    pass
```

Code python

```
tab = [[3, 3, 3], [3, 2, 1], [0, 1, 2]]
for j in range(len(tab[0])):
    print(nombre_occurrences_col(tab, j, 3), end = ' ')
```

Résultat

```
2 1 1
```

3. Écrire de même une fonction `nombre_occurrences_lig`.

Code python

```
def nombre_occurrences_lig(tab, i, e):
    """ [[int]], int, int -> int
    Renvoie le nombre d'occurrences de e dans la ligne i """
```

3 Cherchant à généraliser son système de gestion automatique de notes, un professeur de NSI décide de stocker toutes les notes des devoirs d'une classe dans un tableau 2D. Si un élève est absent à un devoir, on utilise la valeur spéciale `-1`.

Code python

```
trimestre1NSI = [
    ["Alfred",    5, -1,  4,  4, -1],
    ["Brice",    19, 16, -1, 20, 20],
    ["Charline", 16, 14, 15, 16, 18],
    ["David",    10, 11, -1,  9, 10]
]
```

Ainsi, ce professeur a réalisé 5 évaluations au cours de ce trimestre. Alfred a été présent à 3 des 5 évaluations, et a obtenu la note de 5/20 lors de la première évaluation. Charline a été présente à toutes les évaluations et a obtenue sa meilleure note 18/20 lors du dernier devoir. Le premier élément de chaque ligne est toujours le prénom de l'élève. On suppose que tous les prénoms de la classe sont différents.
Dans toutes les questions, les listes `notes` représentent l'ensemble des notes d'une classe. On suppose que les listes `notes` sont de taille au moins 1.

1. Écrire une fonction `nombre_eleves` qui prend en argument une liste `notes` et qui renvoie le nombre d'élèves dont est composée la classe. Si la classe est la liste vide alors la fonction renverra `-1`.

Code python

```
def nombre_eleves(notes):
    """ [[str | int]] -> int
    Renvoie le nombre d'élèves de la classe """
    pass
```

Code python

```
print(nombre_eleves(trimestre1NSI))
```

Résultat

```
4
```

2. Écrire une fonction `nombre_notes` qui prend en argument une liste `notes` et qui renvoie le nombre d'évaluations passées par les élèves de cette classe.

Code python

```
def nombre_notes(notes):
    """ [[str | int ]] -> int
    Renvoie le nombre d'évaluations passées par les élèves """
    pass
```

Code python

```
print(nombre_notes(trimestre1NSI))
```

Résultat

```
5
```

3. **a.** Écrire une fonction `nombre_absences_indice` qui prend en argument une liste `notes`, un indice de ligne `i` et qui renvoie le nombre d'absences de l'élève d'indice `i`.

Code python

```
def nombre_absences_indice(notes, i):
    """ [[str | int]], int -> int
    Renvoie le nombre d'absences de l'élève d'indice i """
    pass
```

Code python

```
for i in range(nombre_eleves(trimestre1NSI)):
    print(nombre_absences_indice(trimestre1NSI, i), end = ' ')
```

Résultat

```
2 1 0 1
```

b. Écrire une fonction `nombre_absences` qui prend en argument une liste `notes`, une chaine de caractère `eleve` et qui renvoie le nombre d'absences de `eleve`. Si `eleve` n'est pas présent dans `notes`, la fonction renverra `-1`.

Code python

```
def nombre_absences(notes, eleve):
    """ [[str | int]], str -> int
    Renvoie le nombre d'absences de eleve """
    pass
```

Code python

```
print(nombre_absences(trimestre1NSI, "Brice"))
```

Résultat

```
1
```

c. Challenge. Lorsque les élèves ont manqué trop d'évaluations, leur moyenne est considérée comme non représentative et ceux-ci doivent passer une épreuve de rattrapage dont le résultat se substitue à la moyenne trimestrielle. Ce professeur décide qu'une moyenne est non représentative quand strictement plus de une évaluation sur 3 a été manquée.

Écrire une fonction `convocation_rattrapage` qui prend en argument une liste `notes` et qui renvoie la liste des prénoms des élèves à convoquer au rattrapage. Il est possible d'écrire et d'utiliser d'autres fonctions intermédiaires si besoin.

Code python

```
def convocation_rattrapage(notes):
    """ [[ str | int ]] -> [str]
    Renvoie la liste des élèves à convoquer au rattrapage """
    pass
```

Code python

```
print(convocation_rattrapage(trimestre1NSI))
```

Résultat

```
['Alfred']
```

4. a. Écrire une fonction `moyenne_devoir` qui prend en argument une liste `notes`, un numéro de devoir (1, 2, ...), et qui renvoie la moyenne de la classe à ce devoir.

Attention. Les élèves absents ne sont pas compabilisés dans le calcul de la moyenne du devoir. On utilisera la fonction `round` pour arrondir au centième de point.

Code python

```
def moyenne_devoir(notes, n):
    """ [[str | int]], int -> int
    Renvoie la moyenne du devoir numero n """
    pass
```

Code python

```
for n in range(1, nombre_notes(trimestre1NSI) + 1):
    print(moyenne_devoir(trimestre1NSI, n), end = ' ')
```

Résultat

```
12.5 13.67 9.5 12.25 16.0
```

En effet, seuls Alfred et Charline étaient présent au troisième devoir. Ils ont obtenu 4/20 et 15/20, la moyenne de la classe à ce devoir est donc de 9,5/20.

b. Écrire une fonction `pire_devoir` qui prend en argument une liste `notes` et qui renvoie le numéro du devoir dont la moyenne est la plus faible. On renverra `-1` si aucun devoir n'est présent dans la liste `notes`.

Code python

```
def pire_devoir(notes):
    """ [[str | int]] -> int
    Renvoie le numéro du devoir dont la moyenne est la plus faible
    """
    pass
```

Code python

```
print(pire_devoir(trimestre1NSI))
```

Résultat

```
3
```

c. Challenge. Lors du calcul de la moyenne trimestrielle, ce professeur utilise la méthode suivante :

- tous les devoirs ont le même coefficient ;
- afin de ne pas pénaliser les élèves, il ne compte pas le devoir le moins bien réussi **par la classe** dans le calcul de la moyenne trimestrielle ;
- la moyenne de chaque élève est calculée par rapport au nombre d'évaluations qu'il a réalisé, non par rapport au nombre d'évaluations totales ;
- si un élève a une moyenne non représentative, la note trimestrielle est de `-1`.

Écrire une fonction `moyennes` qui prend en argument une liste `notes` et qui renvoie la liste des moyennes des élèves, arrondie au centième de point.

Code python

```
def moyennes(notes):
    """ [[str | int]] -> int
    Renvoie la moyenne des élèves de la classe """
    pass
```

Code python

```
print(moyennes(trimestre1NSI))
```

Résultat

```
[['Alfred', -1], ['Brice', 18.75], ['Charline', 16.0], ['David',
 ↪  10.0]]
```