[Q4] On vous fait le tableau

Masquer l'énoncé

On réalise une classe `Matrice` représentant une matrice m x n. Une matrice m x n contient `m` lignes et `n` colonnes, avec une valeur numérique dans chaque cellule.

```
+-----------------------------------+
| Matrice                           |
+-----------------------------------+
| m : nombre de lignes              | # variable d'instance publique
| n : nombre de colonnes            | # variable d'instance publique
| repr : une représentation interne | # variable d'instance publique
+-----------------------------------+
| __init__(self,m,n)                |
| lignes(self)                      |
| colonnes(self)                    |
| get(row,col)                      |
| set(row,col,val)                  |
| __eq__(self,autre)                |
+-----------------------------------+
```

La variable d'instance `repr` contient la représentation interne. Elle est sous la forme d'une liste de triplets de la forme `(row, col, val)`, pour chaque valeur non-nulle `val` à la colonne `col` et la ligne `row`. La représentation est "creuse" : les positions de la matrice ayant une valeur nulle (`0`) peuvent (mais ne doivent pas nécessairement) être omises de la liste; le triplet `(r, c, 0)` peut être présent dans la liste. En particulier, une liste vide représente une matrice dont tous les éléments sont nuls. La liste ne peut contenir au plus qu'une valeur pour une position `(r, c)` donnée; si une nouvelle valeur est assignée elle doit remplacer l'ancienne valeur.

Par exemple, si on exécute le code suivant :

```
matrice = Matrice (2, 3)
matrice.set(1,1,2.0)
matrice.set(1,3,3.0)
matrice.set(2,1,4.0)
matrice.set(1,1,0.0)
```

`matrice` contient la matrice :

```
0.0 0.0 3.0
4.0 0.0 0.0
```

et `matrice.repr` contient la représentation interne :

```
[(1, 3, 3.0), (2, 1, 4.0)]   ou   [(1, 1, 0.0), (1, 3, 3.0), (2, 1, 4.0)]
```

Les spéficications des méthodes de la classe `Matrice` vous sont données ci-dessous comme point de départ. Vous devez compléter l'implémentation de la classe en écrivant le corps des méthodes.

```
__init__(self,m,n) :
    pre: m ≥ 0, n ≥ 0
    post: Initialise une matrice de dimension m x n dont toutes les valeurs sont égales à 0.0
          La variable self.repr contient la représentation.
          Les variables self.m et self.n contiennent, respectivement, les dimensions
          m et n de la matrice.

lignes(self) :
    post: retourne le nombre de lignes m de cette matrice

colonnes(self) :
    post: retourne le nombre de colonnes n de cette matrice

get(self,r,c) :
    pre:  1 <= r <= m
          1 <= c <= n
    post: retourne la valeur de la cellule de la matrice à la ligne r et à la colonne c,
          ou 0.0 si aucune valeur n'a encore été attribuée à cette cellule

set(self,r,c,val) :
    pre:  1 <= r <= m
          1 <= c <= n
    post: assigne la valeur val à la cellule de la matrice
          à la ligne r et à la colonne c

__eq__(self,autre) :
    pre:  /
    post: retourne True si autre est une instance de Matrice de mêmes dimensions et contenant
          les mêmes valeurs que cette matrice, False sinon
```

Conseil : pour pouvoir imprimer des matrices dans vos tests, vous pouvez implémenter aussi la méthode `__str__`, mais ce n'est pas requis dans la question.

On vous rappelle que l'utilisation de `import` n'est pas permise.

Il y a des erreurs dans votre réponse. Votre note est de 94.0%. [Soumission #5ffc44346779dd131aeb355b]

Il semblerait que vous ayez fait des erreurs dans votre code...

```
======================================================================
FAIL: test_matrice_difference (TestQ1.TestQ)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/usr/local/lib/python3.6/site-packages/timeout_decorator/timeout_decorator.py", line 81, in new_function
    return function(*args, **kwargs)
  File "./src/TestQ1.py", line 319, in test_matrice_difference
AssertionError: True != False : Votre méthode eq ne renvoie pas False en comparant une matrice et un str.

======================================================================
FAIL: test_matrice_different_taille (TestQ1.TestQ)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/usr/local/lib/python3.6/site-packages/timeout_decorator/timeout_decorator.py", line 81, in new_function
    return function(*args, **kwargs)
  File "./src/TestQ1.py", line 355, in test_matrice_different_taille
AssertionError: True != False : Votre méthode eq ne renvoie pas False en comparant deux matrices différentes de mêmes dimensions.

----------------------------------------------------------------------
Ran 15 tests in 0.030s

FAILED (failures=2)
```

Question 1: Matrice

Complétez la définition des méthodes dans la classe `Matrice`.

```
class Matrice :

    def __init__(self,m,n) :
        """
        pre: m ≥ 0, n ≥ 0
        post: Initialise une matrice de dimension m x n dont toutes les valeurs sont égales à 0.0
              La variable self.repr contient la représentation.
              Les variables self.m et self.n contiennent, respectivement, les dimensions
              m et n de la matrice.
        """
        self.m = m
        self.n = n
        self.repr = []

    def lignes(self) :
        """
        pre:  /
        post: retourne le nombre de lignes m de cette matrice
        """
        return self.m

    def colonnes(self) :
        """
        pre:  /
        post: retourne le nombre de colonnes n de cette matrice
        """
        return self.n

    def get(self,r,c) :
        """
        pre:  1 <= r <= m
              1 <= c <= n
        post: retourne la valeur de la cellule de la matrice à la ligne r et à la colonne c,
              ou 0.0 si aucune valeur n'a encore été attribuée à cette cellule
        """
        for i in self.repr:
            if i[0] == r and i[1] == c:
                return i[2]
        return 0.0

    def set(self,r,c,val) :
        """
        pre:  1 <= r <= m
              1 <= c <= n
        post: assigne la valeur val à la cellule de la matrice
              à la ligne r et à la colonne c
        """
        count = 0
        for i in range(len(self.repr)):
            if self.repr[i][0] == r and self.repr[i][1] == c:
                count += 1
                self.repr[i] = (self.repr[i][0],self.repr[i][1],val)
        if count == 0:
            self.repr.append((r, c, val))

    def __eq__(self,autre) :
        """
        pre:  /
        post: retourne True si autre est une instance de Matrice de mêmes dimensions et contenant
              les mêmes valeurs que cette matrice, False sinon
        """
        if isinstance(autre,Matrice) and self.m == autre.m and self.n == autre.n:
            for i in self.repr:
                count = 0
                for j in autre.repr:
                    if i == j or i[2] == 0.0 or j[2] == 0.0:
                        count += 1
                if count == 0:
                    return False
        return True

    def __str__(self):
        return self.repr
```

Question 2: Zone de test

Insérez votre code de test optionnel ci-dessous. Vous pouvez utiliser `print` pour tester votre programme. Votre code de test sera exécuté à la suite de la définition de la classe `Matrice` (il ne faut pas la recopier ici).

Exemple:

```
matrice = Matrice (2, 3)
matrice.set(1, 1, 2.0)
matrice.set(1, 3, 3.0)
matrice.set(2, 1, 4.0)
print(matrice.repr)
matrice.set(1, 1, 0.0)

print(matrice.get(1, 1))  # 0.0
print(matrice.get(1, 3))  # 3.0
print(matrice.repr)
matrice.set(1, 1, 3.0)

print(matrice.repr)
# [(1, 3, 3.0), (2, 1, 4.0)] ou [(1, 1, 0.0), (1, 3, 3.0), (2, 1, 4.0)]

# VOS TESTS ICI
```

La date limite est dépassée, vous ne pouvez plus soumettre