BACCALAUREAT

SESSION 2022

Épreuve de l'enseignement de spécialité

NUMERIQUE et SCIENCES INFORMATIQUES

Partie pratique

Classe Terminale de la voie générale

Sujet n°20

DUREE DE L'EPREUVE : 1 heure

Le sujet comporte 2 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)

L'opérateur « ou exclusif » entre deux bits renvoie 0 si les deux bits sont égaux et 1 s'ils sont différents : $0 \oplus 0 = 0$, $0 \oplus 1 = 1$, $1 \oplus 0 = 1$, $1 \oplus 1 = 0$
On représente ici une suite de bits par un tableau contenant des 0 et des 1.

Exemples :

```
a = [1, 0, 1, 0, 1, 1, 0, 1]
b = [0, 1, 1, 1, 0, 1, 0, 0]
c = [1, 1, 0, 1]
d = [0, 0, 1, 1]
```

Écrire la fonction `xor` qui prend en paramètres deux tableaux de même longueur et qui renvoie un tableau où l'élément situé à position `i` est le résultat, par l'opérateur « ou exclusif », des éléments à la position `i` des tableaux passés en paramètres.

En considérant les quatre exemples ci-dessus, cette fonction doit passer les tests suivants :

```
assert(xor(a, b) == [1, 1, 0, 1, 1, 0, 0, 1])
assert(xor(c, d) == [1, 1, 1, 0])
```

EXERCICE 2 (4 points)

Dans cet exercice, on appelle carré d'ordre n un tableau de n lignes et n colonnes dont chaque case contient un entier naturel.

Exemples :

1	1
1	1

c2

Un carré d'ordre 2

2	9	4
7	5	3
6	1	8

c3

Un carré d'ordre 3

4	5	16	9
14	7	2	11
3	10	15	6
13	12	8	1

c4

Un carré d'ordre 4

Un carré est dit magique lorsque les sommes des éléments situés sur chaque ligne, chaque colonne et chaque diagonale sont égales. Ainsi c2 et c3 sont magiques car la somme de chaque ligne, chaque colonne et chaque diagonale est égale à 2 pour c2 et 15 pour c3. c4 n'est pas magique car la somme de la première ligne est égale à 34 alors que celle de la dernière colonne est égale à 27.

La classe `Carre` ci-après contient des méthodes qui permettent de manipuler des carrés.

Compléter la fonction `est_magique` qui prend en paramètre un carré et qui renvoie la valeur de la somme si ce carré est magique, `False` sinon.

```
class Carre:
    def __init__(self, tableau = [[]]):
        self.ordre = len(tableau)
        self.valeurs = tableau

    def affiche(self):
        '''Affiche un carré'''
        for i in range(self.ordre):
            print(self.valeurs[i])

    def somme_ligne(self, i):
        '''Calcule la somme des valeurs de la ligne i'''
        return sum(self.valeurs[i])

    def somme_col(self, j):
        '''Calcule la somme des valeurs de la colonne j'''
        return sum([self.valeurs[i][j] for i in range(self.ordre)])

def est_magique(carre):
    n = carre.ordre
    s = carre.somme_ligne(0)

    #test de la somme de chaque ligne
    for i in range(..., ...):
        if carre.somme_ligne(i) != s:
            return ...

    #test de la somme de chaque colonne
    for j in range(n):
        if ... != s:
            return False

    #test de la somme de chaque diagonale
    if sum([carre.valeurs[...][...] for k in range(n)]) != s:
            return False
    if sum([carre.valeurs[k][n-1-k] for k in range(n)]) != s:
            return False

    return ...
```

Tester la fonction `est_magique` sur les carrés `c2`, `c3` et `c4`.

```python
"""
Author: Pascal Padilla
Source: correction de l'exercice 1 du sujet 20 des épreuves pratiques NSI 2022
"""


from doctest import testmod


def xor(tab1: list[int], tab2: list[int]) -> list[int]:
    """
    La fonction xor qui prend en paramètres deux tableaux
    de même longueur et qui renvoie un tableau où l¿élément
    situé à position i est le résultat, par l¿opérateur
    « ou exclusif », des éléments à la position i des
    tableaux passés en paramètres

    Args:
        tab1 (list[int]): tableau de 0 et de 1
        tab2 (list[int]): tableau de 0 et de 1 (même taille que tab1)

    Returns:
        list[int]: XOR entre chaque élément i de tab1 et tab2

    Tests et exemples:
    >>> a = [1, 0, 1, 0, 1, 1, 0, 1]
    >>> b = [0, 1, 1, 1, 0, 1, 0, 0]
    >>> xor(a, b)
    [1, 1, 0, 1, 1, 0, 0, 1]
    >>> c = [1, 1, 0, 1]
    >>> d = [0, 0, 1, 1]
    >>> xor(c, d)
    [1, 1, 1, 0]
    """
    nb_bits = len(tab1)

    # création du tableau résultat
    # (pour l'instant plein de vide)
    tab_xor = [None] * nb_bits

    # parcours des tableaux tab1 et tab2 bit par bit
    for i in range(nb_bits):
        bit1 = tab1[i]
        bit2 = tab2[i]

        # pour éviter de faire 4 tests, on remarque que
        # XOR vaut 0 <=> bit1 et bit2 sont différents !
        if bit1 == bit2:
            tab_xor[i] = 0
        else:
            tab_xor[i] = 1

    return tab_xor


a = [1, 0, 1, 0, 1, 1, 0, 1]
b = [0, 1, 1, 1, 0, 1, 0, 0]
c = [1, 1, 0, 1]
d = [0, 0, 1, 1]


# tests avec des affichages
print(xor(a, b))
print(xor(c, d))


# tests avec des assertions
assert(xor(a, b) == [1, 1, 0, 1, 1, 0, 0, 1])
assert(xor(c, d) == [1, 1, 1, 0])


# tests de la fonction avec doctest
testmod()
```

```
"""
Author: Pascal Padilla
Source: correction de l'exercice 2 du sujet 20 des épreuves pratiques NSI 2022

Remarque:
    la classe Carre a 2 propriétés et 3 méthodes:
        ¿ self.ordre: ordre du carré (et donc nb de ligne et colonne)
        ¿ self.valeur: tableau contenant les valeurs (tableau de tableau)

    il est difficile de faire les tests car il faut créer les
    carrés c2, c3 et c4 avec la POO
"""


class Carre:
    def __init__(self, tableau = [[]]):
        self.ordre = len(tableau)
        self.valeurs = tableau

    def affiche(self):
        '''Affiche un carré'''
        # affiche ligne par ligne le tableau de valeurs
        for i in range(self.ordre):
            print(self.valeurs[i])

    def somme_ligne(self, i):
        '''Calcule la somme des valeurs de la ligne i'''
        # utilise la fonction `sum` qui calcule la somme d'un tableau
        return sum(self.valeurs[i])

    def somme_col(self, j):
        '''calcule la somme des valeurs de la colonne j'''
        # génère un tableau par compréhension pour créer la colonne j
        # puis utilise la fonction sum pour calculer la somme des valeurs
        return sum([self.valeurs[i][j] for i in range(self.ordre)])

def est_magique(carre):
    n = carre.ordre
    s = carre.somme_ligne(0)

    #test de la somme de chaque ligne
    # ¿ de la 2ème ligne à la (n - 1)ième ligne
    for i in range(1, n):
        if carre.somme_ligne(i) != s:
            return False

    #test de la somme de chaque colonne
    # ¿ utilise la méthode .somme_col(j) pour calculer la somme de la colonne
    for j in range(n):
        if carre.somme_col(j) != s:
            return False

    #test de la somme de chaque diagonale
    # ¿ création d'un tableau en compréhension : tab[k][k] for k in ...
    #   pour avec le tableau de la diagonale
    #   [tab[0][0], tab[1][1], tab[2][2], ...]
    if sum([carre.valeurs[k][k] for k in range(n)]) != s:
        return False
    if sum([carre.valeurs[k][n-1-k] for k in range(n)]) != s:
        return False

    # si aucun renvoie n'a eu lieu, c'est le carré est magique
    return True


# tests
c2 = Carre([[1, 1], [1, 1]])
print(est_magique(c2))

c3 = Carre([[2, 9, 4], [7, 5, 3], [6, 1, 8]])
print(est_magique(c3))

c4 = Carre([[4, 5, 16, 9], [14, 7, 2, 11], [3, 10, 15, 6], [13, 12, 8, 1]])
```

```
print(est_magique(c4))
```