Devoir surveillé n°2 – NSI
09/10/2023

Exercice 1 (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.

Exemple

1	7
7	1

`c2`
Un carré d'ordre 2

3	4	5
4	4	4
5	4	3

`c3`
Un carré d'ordre 3

2	9	4
7	0	3
6	1	8

`c3bis`
Un autre carré d'ordre 3

Un carré est dit **semimagique** lorsque les sommes des éléments situés sur chaque ligne et chaque colonne sont égales.

- Ainsi `c2` et `c3` sont semimagiques car la somme de chaque ligne et chaque colonne est égale à 8 pour `c2` et 12 pour `c3`.
- Le carré `c3bis` n'est pas semimagique car la somme de la première ligne est égale à 15 alors que celle de la deuxième ligne est égale à 10.

La classe `Carre` en page suivante contient des méthodes qui permettent de manipuler des carrés :

- La méthode `constructeur` crée un carré sous forme d'un tableau à deux dimensions à partir d'une liste d'entiers, et d'un ordre.
- La méthode `affiche` permet d'afficher le carré créé.

Exemple :

```
>>> liste = (3, 4, 5, 4, 4, 4, 5, 4, 3)
>>> c3 = Carre(liste, 3)
>>> c3.affiche()
[3, 4, 5]
[4, 4, 4]
[5, 4, 3]
```

Recopier et compléter la méthode `est_semimagique` ci-après qui renvoie `True` si le carré est semimagique, `False` sinon.

```
class Carre:
    def __init__(self, liste, n):
        self.ordre = n
        self.tableau = [[liste[i + j * n] for i in range(n)]
                        for j in range(n)]
    def affiche(self):
        '''Affiche un carré'''
        for i in range(self.ordre):
            print(self.tableau[i])

    def somme_ligne(self, i):
        '''Calcule la somme des valeurs de la ligne i'''
        somme = 0
        for j in range(self.ordre):
            somme = somme + self.tableau[i][j]
        return somme
```

```
    def somme_col(self, j):
        '''Calcule la somme des valeurs de la colonne j'''
        somme = 0
        for i in range(self.ordre):
            somme = somme + self.tableau[i][j]
        return somme

    def est_semimagique(self):
        s = self.somme_ligne(0)

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

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

        return ...
```

Exercice 2 (6 points)

Dans le tableau ci-dessous, on donne les caractéristiques nutritionnelles, pour une quantité de 100 grammes, de quelques aliments.

	Lait entier UHT	Farine de blé	Huile de tournesol
Énergie (kcal)	65.1	343	900
Protéine (grammes)	3.32	11.7	0
Glucides (grammes)	4.85	69.3	0
Lipides (grammes)	3.63	0.8	100

Pour chaque aliment, on souhaite stocker les informations dans un objet de la classe `Aliment` définie ci-dessous, où `e`, `p`, `g` et `l` sont de type `float` et désignent respectivement les quantités d'énergie, de protéines, de glucides et de lipides de l'aliment.

```
class Aliment:
    def __init__(self, e, p, g, l):
        self.energie = e
        self.proteines = p
        self.glucides = g
        self.lipides = l
```

1. (a) Écrire, à l'aide du tableau des caractéristiques nutritionnelles, l'instruction en langage Python pour instancier l'objet `lait`.

   (b) Donner l'instruction qui permet d'obtenir la valeur 65.1 de l'objet `lait` instancié dans la question précédente.

   (c) En fait, la masse de protéines dans le lait est 3.4 au lieu de 3.32.

   Donner l'instruction qui modifie la masse de protéines de l'objet lait instancié dans la question 1a.

2. On souhaite ajouter une méthode `energie_reelle` à la classe `Aliment` qui calcule l'énergie en kcal d'un aliment en fonction d'une masse donnée.

   Exemple Pour 245 grammes de lait, l'énergie réelle sera $245 \times 65.1 \div 100 = 159.495$ kcal. L'instruction `lait.energie_reelle(245)` renvoie alors `159.495`.

   Recopier et compléter les lignes dans la méthode ci-dessous :

```
    def energie_reelle(...,masse):
        return ...
```

3. On regroupe les caractéristiques nutritionnelles du tableau dans le dictionnaire suivant, les clés étant des chaînes de caractères donnant le nom de l'aliment et les valeurs associées des objets de la classe Aliment :

```
nutrition = {'lait' : Aliment(65.1,3.4,4.85,3.63),
             'farine' : Aliment(343,11.7,69.3,0.8),
             'huile' : Aliment(900,0,0,100)
            }
```

   (a) Donner l'instruction qui permet d'obtenir la valeur énergétique en kcal du lait à partir des données de ce dictionnaire.

   (b) Donner l'instruction qui permet d'obtenir la valeur énergétique réelle de 220 grammes de lait à partir des données de ce dictionnaire.

4. Une recette de gâteau (sans œuf) utilise les ingrédients suivants :
   - 230 g de farine ;
   - 220 g de lait ;
   - 100 g d'huile.

   Les quantités d'ingrédients, en grammes, sont regroupées dans le dictionnaire suivant :

```
recette_gateau={'lait' : 220, 'farine' : 230, 'huile' : 100}
```

   Écrire, en utilisant la méthode `energie_reelle` de la classe `Aliment`, une définition en Python de la fonction `calcul_energie_tot(recette, nutrition)` (hors de la classe) qui calcule et retourne l'énergie réelle totale d'une `recette` étant données les informations de `nutrition`.

   On doit alors par exemple obtenir, dans la console :

```
>>> calcul_energie_tot(recette_gateau, nutrition)
1832.12
```