NSI Première

Travaux dirigés : tableaux à deux dimensions

qkzk

2021/05/23

PDF pour impression

Tableaux à deux dimensions

1. Coordonnées dans une grille

On considère une grille rectangulaire représentée en mémoire par un tableau à deux dimension :

```
grille = [[1, 2, 1, 5],
          [2, 1, 4, 3],
          [1, 4, 5, 1]]
```

Chaque élément de la grille est un entier.

L'algorithme, incomplet, qui suit permet de cumuler les éléments de la grille :

```
somme = 0
pour chaque ligne de la grille faire :
    pour chaque cellule d'une ligne faire :
        COMPLÉTER
    fin du pour
fin du pour
```

1. Compléter l'algorithme
2. Combien d'additions sont réalisées lorsqu'on applique l'algorithme à la grille donnée en exemple ? Pour une grille contenant L lignes et C colonnes ?
3. Traduire cet algorithme dans une fonction python `cumul_grille`

2. Parcourir un tableau à deux dimensions.

On considère un tableau à deux dimensions enregistré dans `grille` :

```
grille = [[4, 5, 2, 1, 3],
          [5, 1, 3],
          [7, 8, 1, 2],
          [0, 1, 2, 5]]
```

Remarquons d'abord que les sous listes sont de taille différente.

1. Proposer un algorithme permettant de compter les éléments de la grille.
2. Traduire cet algorithme en Python.
3. Adapter votre algorithme pour calculer aussi la somme de tous les éléments en une seul parcours.
4. Traduire cette modification dans votre programme Python.
5. Écrire une fonction `moyenne_grille` qui prend en paramètre une grille comme la précédente et renvoie la valeur moyenne des éléments qu'elle contient.

3. D'une liste plate à une double liste et inversement.

On considère un tableau à deux dimensions comme celui ci-dessous :

2	3	5
5	2	7
4	3	1
6	2	9

1. a) Construire un tableau à une dimension comportant chaque élément du tableau. Les lignes sont simplement mises bout à bout.

   b) Combien d'éléments comporte cette liste ?

   c) On souhaite accéder à l'élément 7 du tableau ci-dessus dans la liste.

   7 est situé à la deuxième ligne, troisième colonne du tableau ci-dessus.

   On compte les éléments dans le tableau à 2 dimensions à partir de l'indice 0.

   - Son numéro de ligne est `no_ligne = 1`
   - son numéro de colonne est `no_colonne = 2`
   - chaque ligne comporte `len_ligne = 3` éléments

   Son indice dans la liste à une dimension est donc.

   `indice = len_ligne * no_ligne + no_colonne` qui vaut `1 * 3 + 2 = 5`

   Calculer l'indice dans la liste des éléments 6 et 9.

2. Nous allons transformer cette liste en une liste à deux dimensions.

   L'algorithme est le suivant :

```
len_ligne   : un entier,
liste_plate : une liste à 1 dimension
liste_2d    : une liste vide
Pour i allant de 0 à longueur de la liste - 1:
    si i % len_ligne == 0:
      si i != 0:
        ajouter ligne à liste_2d
      créer une liste vide : ligne
    ajouter à ligne l'élément d'indice i de liste_plate
ajouter ligne à liste_2d
```

   Le résultat final est une liste à deux dimensions comme celle-ci :

```
[[2, 3, 5],
 [5, 2, 7],
 [4, 3, 1],
 [6, 2, 9]]
```

   Écrire cet algorithme en Python.

3. Dans l'autre sens.

   On part cette fois d'une liste à deux dimensions comme :

```
liste_2d = [[3, 5, 1], [4, 7, 9], [6, 2, 3], [1, 2, 1]]
```

   a. Écrire un algorithme permettant d'enregistrer tous les nombres dans une liste à une dimension.

   b. Écrire le programme Python correspondant.

4. Élément extrême

On a relevé dans un tableau appelé `temperatures` les températures à midi de chaque jour de la semaine.

Exemple : `temperatures = [12, 10, 14, 11, 13, 16, 12]`

Ces relevés hebdomadaires ont ensuite été regroupés dans un tableau `releves`

Exemple :

```
releves = [[12, 10, 14, 11, 13, 16, 12],
           [10, 13, 15, 10, 11, 11, 15],
           ...
           [11, 12, 13, 12, 14, 11, 14]]
```

On souhaite connaître le numéro de la semaine et le jour de la semaine durant lequel la température a été maximale, minimale.

Voici la proposition de Jérôme :

```
temp_max = 0
for semaine in releves:
    for temperature in semaine:
        if temperature > temp_max:
            temp_max = temperature
return temperature
```

Lorsqu'il exécute son script, Jérôme obtient une erreur :

```
syntax error: 'return' outside a function
```

1. D'après le message d'erreur, quel est le problème ? Rectifiez le code afin qu'il ne lève plus cette erreur.
2. L'algorithme de Jérôme répond il au problème ? Quelle sera l'information obtenue ?
3. Proposez une correction du programme afin d'obtenir l'information souhaitée .

5. Modifier un tableau de pixels

Le programme suivant génère une image rectangulaire comportant un dégradé du blanc au noir :

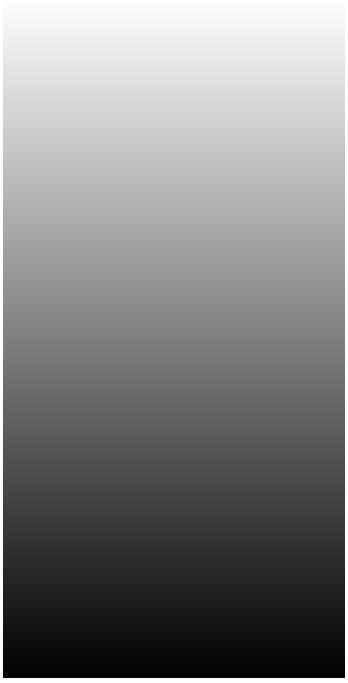
```python
from PIL import Image


def degrade_blanc_noir():
    # nouvelle image, 255 de large, 128 de haut
    img = Image.new('RGB', (255, 128))
    # on charge la matrice des pixels
    pixels = img.load()

    for x in range(255):
        for y in range(128):
            # attention à la notation [x, y] !!!
            pixels[x, y] = (255 - x, 255 - x, 255 - x)

    img.show()  # afficher dans la console
```

1. Adaptez le code de cette fonction pour générer un dégradé du noir au blanc.
2. On souhaite tourner l'image afin de produire un dégradé vertical. Adaptez la fonction.



6. Matrice enregistrée en ligne

On a enregistré le nombre de naissances dans une petite maternité pour chaque jour de la semaine. Certains langages ne permettent pas de construire des listes de listes. À la suite d'un enregistrement en mémoire, le tableau ci-dessous a été enregistré dans une longue liste Python :

Lundi	Mardi	Mercredi	Jeudi	Vendredi
1	3	2	3	2
2	4	2	6	4
1	3	2	1	4

```python
naissances = [1, 3, 2, 3, 2, 2, 4, 2, 6, 4, 1, 3, 2, 1, 4]
```

Lorsqu'on manipule de telles données, il est courant d'utiliser 2 indices :

```
for i in range(3):
    for j in range(5):
        jour = 5 * i + j
        nb_naissance = naissances[jour]
        print(i, j, nb_naissance)
```

1. Faire tourner le programme précédent à la main et écrire les affichages qu'il produit.
2. On souhaite créer un programme qui affiche, pour n'importe quelle liste Python de longueur $n \times p$ ses élements par bloc de taille p.

   Le programme précédent est un exemple pour des blocs de taille 3×5.

   Proposer une fonction qui :

   a. prend en entrée une liste de taille $n \times p$ et les nombres entiers n et p ;

   b. ne retourne rien ;

   c. affiche les éléments par bloc ainsi :

   ```
   1 3 2 3 2
   2 4 2 6 4
   1 3 2 1 4
   ```

   On utilisera le paramètre *positionnel* de la fonction `print` :

   ```
   >>> for x in range(3):
   ...     print(x, end=" ")
   ...
   0 1 2
   ```

   Par défaut, Python ajoute un retour à la ligne à la fin d'un `print` : `end="\n"`
3. Écrire un programme Python `transforme_liste(liste, n, p)` qui transforme une telle liste de taille $n \times p$ en un tableau à deux dimensions de n lignes et p colonnes :

   ```
   >>> transforme_liste(naissances, 3, 5)
   [[1, 3, 2, 3, 2],
    [2, 4, 2, 6, 4],
    [1, 3, 2, 1, 4]]
   ```

Listes par compréhension

7. Lire des listes par compréhension

Décrire les listes suivantes :

1. `carres = [x ** 2 for x in range(5)]`
2. `modulo_2 = [x for x in range(10) if x % 3 == 2]`
3. On considère les mots :

   ```
   mots = ["bonjour", "manipuler", "avant", "mercredi", "parcours"]

   mot_avec_a  = [mot for mot in mots if 'a' in mot]
   longueurs = [len(mot) for mot in mots]
   ```

6. Construire une liste par compréhension

1. Construire par compréhension la liste des cubes des entiers entre 3 et 10.
2. Les joueurs de l'équipe sont enregistrés dans un tableau :

```
equipe = [('Marcel', 'attaquant'), ('Pierre', 'défenseur'), ('Joseph', 'Milieu'),
          ('Guy', 'attaquant'), ('Raoul', 'défenseur'), ('Gérard', 'Milieu')]
```

   Construire par compréhension la liste des attaquants de l'équipe.

3. Dans la variable `nombres` on a enregistré une liste d'entiers naturels.

   Construire par compréhension la liste des entiers multiples de 3 parmi `nombres`.

Itérer sur un Dictionnaire

8. Les développeurs

On considère le statut des membres d'une équipe de développeurs :

```
developpeurs = {
        'Marcel': 'technicien',
        'Fanny': 'ingénieur',
        'Paul': 'ingénieur',
        'Frank': 'technicien',
        ...}
```

La fonction `alerter(nom, message)` envoie un message à un individu donné par son nom. Ses paramètres d'entrée sont des chaînes de caractères.

1. Écrire une boucle qui parcourt le dictionnaire et alerte tous les ingénieurs
2. Écrire une boucle qui parcourt le dictionnaire et compte les ingénieurs.
3. Comment récupérer la liste des prénoms de l'équipe de développeurs ?

9. Les monstres du jeu

On a enregistré dans un dictionnaires les caractéristiques des monstres qui figurent dans un jeu :

```
monstres = {
    'Grogneur': ['Tempête', 12, 14],
    'Frappé':   ['Tempête', 8, 16],
    'Brûleur':  ['Feu' 20, 6],
    'Givré':    ['Glace', 17, 13],
    'Charbon':  ['Feu', 10, 18]
    ...}
```

Le premier élément de chaque liste est le type du monstre, ensuite sont indiqués sa force et sa vie.

1. Créer, à l'aide d'une boucle `for` Python, la liste des noms des monstres.
2. Proposer une boucle Python qui affiche successivement tous les types des monstres :

```
Tempête
Tempête
Feu
Glace
Feu
...
```

3. Comment déterminer le monstre qui a le plus de vie ? Le moins de force ?

   Englober dans une fonction :

```
>>> le_moins_fort(monstres)
Frappé
>>> le_plus_de_vie(monstres)
Charbon
```