TP 1

Révisions Programmation

25 septembre 2024

Créez un dossier à votre nom, et à l'intérieur un dossier `infoPSI`. Enregistrez un fichier `TP1.py` dans votre dossier `infoPSI`. Faire **tous les exercices dans ce fichier** (à chaque exercice va correspondre une fonction). Lorsque vous testez une fonction, mettre les appels à la fonction directement **dans votre programme** puis mettre les appels en commentaire avec `#` avant de passer à l'exercice suivant (**ne pas les effacer**). A la fin du TP, vous devez rendre votre fichier `TP1.py` en vous identifiant sur le site `https://cahier-de-prepa.fr/psi-lessouriau`

Important :

— **respectez les noms** qu'on vous demande de donner aux fonctions.
— testez vos fonctions pour les vérifier **avec plusieurs valeurs pertinentes !**
— **laissez en commentaire au moins un appel par fonction**

Exercice 0 : Compte à rebours

Écrivez une fonction `rebours` qui prend en argument un entier n et affiche en ordre décroissant les entiers compris entre n et 0.

Exercice 1 : Mult3pas9

Écrire une fonction `mult3pas9` qui prend en argument un entier n et renvoie `True` si n est multiple de 3 mais pas de 9, et `False` sinon.
Votre fonction ne doit comporter qu'une seule instruction (une seule ligne de code, **pas de** `if`).

Exercice 2 : Produit

Écrire une fonction `produit` qui lit des entiers entrés par l'utilisateur jusqu'à ce que l'utilisateur entre 0, puis qui affiche le produit des entiers non nuls entrés par l'utilisateur. Ne pas utiliser de liste ! Tester votre fonction. Exemple d'exécution :

```
Entrez les nombres à multiplier en terminant par 0 :
4
3
0
Le produit vaut 12.0
```

Exercice 3 : Triangle

Écrire une fonction `triangle` qui prend en argument un entier n, puis qui affiche le triangle d'entiers de taille n. Ne pas faire d'appel à la fonction `compter` (la fonction `triangle` doit faire elle-même tous ses affichages). Par exemple, l'exécution de `triangle(4)` doit donner :

```
1
12
123
1234
```

Faire de même une fonction `triangle2` telle que le triangle obtenu pointe vers le bas :

```
1234
123
12
1
```

Exercice 4 : Suite

Écrire une fonction `terme` qui prend en argument un entier n et qui renvoie le terme d'indice n de la suite définie par récurrence par $u_0 = 1$, $u_1 = 2$ et $u_n = 3 * u_{n-1} + u_{n-2}$ pour $n \geq 2$. Par exemple, `terme(4)` vaut 76.

Exercice 5 : Miroir

Écrivez une fonction `miroir` qui prend en argument une liste ℓ et renvoie une nouvelle liste dont les éléments sont les éléments ℓ mais dans l'ordre inverse de celui dans lequel ils sont dans ℓ. Ne pas utiliser la méthode `reverse` de Python : le but est de faire comme si vous êtes le programmeur qui écrit cette méthode.

Exercice 6 : TestTri

Écrivez une fonction `testTri` qui prend comme argument une liste d'entiers et teste si cette liste est triée en ordre croissant.

Exercice 7 : Incrémente

Écrivez une fonction `incremente` qui prend en argument un tableau bidimensionnel (c'est-à-dire une liste de listes) contenant des entiers, et qui ajoute 1 à chaque entier du tableau puis le renvoie.

Exercice 8 : Table

Écrivez une fonction `table` qui prend en argument deux entiers n et m, et qui renvoie un tableau à deux dimensions (c'est-à-dire une liste de listes) de n lignes et de m colonnes, dont la case d'indice i, j contient le produit $i \times j$ pour tous i, j.

Exercice 9 : Recherche

Écrivez une fonction `recherche` qui prend en argument un tableau bidimensionnel (c'est-à-dire une liste de listes) et une valeur et teste si cette valeur apparaît dans une des cases du tableau.

Exercice 10 : Carré magique

Un carré magique est une matrice carrée telle que la somme de chaque rangée, de chaque colonne et de chaque diagonale a la même valeur. Un carré magique de n lignes est dit normal s'il contient chaque entier compris entre 1 et n^2 exactement une fois. Par exemple, le tableau suivant est un carré magique normal :

6	7	2
1	5	9
8	3	4

Dans la suite on écrit des fonctions pour tester si un tableau bidimensionnel est un carré magique normal.

1. Écrivez une fonction `carre` qui teste si une liste de listes d'entiers donnée en paramètre est une matrice carrée (c'est-à-dire un tableau dont toutes les lignes et les colonnes ont même longueur).
2. Écrivez deux fonctions `sommeLigne` et `sommeColonne` qui prennent en argument un entier i et une matrice (pas forcément carrée) et qui renvoient : pour `sommeLigne`, la somme des éléments de la ie ligne de la matrice, et pour `sommeColonne`, la somme des éléments de la ie colonne de la matrice.
3. Écrivez deux fonctions `sommeDiagonaleMajeure` et `sommeDiagonaleMineure` qui prennent en argument une matrice carrée et renvoient respectivement la somme de la diagonale majeure, et celle de la diagonale mineure du tableau passé en paramètre.
4. Écrivez une fonction `carreMagique` qui prend en argument un tableau bidimensionnel, et teste s'il s'agit d'un carré magique (pas forcément normal).
5. Pour savoir si un carré magique est normal, il faut compter le nombre de fois qu'apparaissent ses éléments. Écrivez une fonction `histogramme` qui prend en argument un tableau bidimensionnel t et qui renvoie l'histogramme du tableau t, c'est-à-dire la liste h de taille n^2 (avec n le nombre de lignes de t) tel que pour tout $i < n^2$, $h[i]$ contient le nombre d'occurrences de la valeur $i+1$ dans le tableau t (on peut aussi choisir de construire h de taille $n^2 + 1$ afin que pour $1 \leq i \leq n^2$, $h[i]$ contient le nombre d'occurrences de la valeur i dans le tableau t). Par exemple pour un carré magique 3×3, son histogramme aura 9 cases, pour compter le nombre d'apparition des éléments compris entre 1 et 9.
6. Écrivez une fonction `carreMagiqueNormal` qui prend en argument un tableau bidimensionnel, et teste s'il s'agit d'un carré magique normal.