LISTE PYTHON - EXERCICES

1. Exercices élémentaires

1 : création d'une liste

Créer une liste avec les entiers de 1 à 10

Créer une liste avec les entiers de 20 à 30

Créer une liste avec les carrés de 1 à 10 (1, 4, 9, 16, … 100)

Créer une liste avec 10 entiers pris au hasard entre 1 et 100

Pour chaque exercice, faire une version avec une fonction qui retourne la liste.

Afficher le contenu de la liste.

2 : somme des éléments d'une liste

Calculer la somme des éléments d'une liste : faites ça avec une fonction python. Puis faites ça « à la main », sans fonction python avec uniquement des tests et des boucles.

Mettez votre algorithme dans une fonction qui retourne la somme.

3 : chercher une valeur dans d'une liste

Cherchez une valeur dans une liste : faites ça avec une fonction python. Puis faites ça « à la main », sans fonction python avec uniquement des tests et des boucles.

Mettez votre algorithme dans une fonction qui retourne l'index de la valeur cherchée.

4 : Slicing

Soit la liste contenant des entiers de 0 à 30 :

Extraire les 5 premiers

Extraire des 5 derniers

Extraire uniquement les nombres impairs

Extraire les nombres de 10 à 20

Extraire les multiples de 3 de 10 à 20

Produire une liste en sens inverse

Produire une liste de multiples de 5 de 25 à 5 en partant de cette liste.

Ajoutez 20.5 au bon endroit

Ajouter 19.5, 20.5 et 21.5 au bon endroit en une seule opération en modifiant la liste

Reproduisez la liste de départ.

5 : valeur la plus grande d'une liste

Cherchez la valeur la plus grande d'une liste : faites ça avec une fonction python. Puis faites ça « à la main », sans fonction python avec uniquement des tests et des boucles.

Mettez votre algorithme dans une fonction qui retourne la valeur la plus grande.

6 : retournement d'un tableau

Écrire une fonction qui retourne une liste « inversée » : le premier élément est permuté avec le dernier, le deuxième avec l'avant dernier, etc.

On fera ça avec des append() puis avec les slices.

7 : décalage vers le bas

Écrire une fonction qui retourne une liste « décalée vers le bas (le 1er élément passe en 2ème, le 2ème en 2ème, etc… et le dernier passe en premier.

On fera ça avec des append() puis avec les slices.

8 : décalage vers le haut

Comme le précédent, mais en décalant vers le haut : le premier passe en dernier.

9 : suppression d'un élément

Supprimer une valeur dans d'une liste : faites ça avec une fonction python. Puis faites ça « à la main », sans fonction python avec uniquement des tests et des boucles.

Mettez votre algorithme dans une fonction qui retourne la liste mise modifiée.

10 : suppression des doublons

Supprimer tous les doublons d'une liste : faites ça avec une technique spécifique du python. Puis faites ça « à la main », sans fonction python avec uniquement des tests et des boucles.

Mettez votre algorithme dans une fonction qui retourne la liste mise modifiée.

2. Exercices avancés

Exercice 1 : recherche d'un élément dans un tableau trié

Écrire une fonction qui recherche une valeur dans un tableau trié sans doublon. On utilisera la méthode de recherche dichotomique qui consiste à chercher au milieu du tableau, puis à restreindre le tableau à un sous tableau plus petit (par exemple, si je cherche entre 0 et 1000, j'essaye 500, si c'est plus petit, j'essaye entre 0 et 500, donc 250, si c'est plus grand, j'essaye entre 250 et 500 donc 375, etc.).

Tester votre fonction sur un tableau de 1_000_000 d'entiers.

Votre fonction retournera l'indice du nombre cherché et aussi le nombre de tests effectués pour trouver, ou pas, le nombre cherché.

Testez la vitesse d'exécution de votre fonction en la comparant la celle de la fonction standard de recherche en python : que constatez-vous ?

Pour calculer le temps de calcul on fait :

```
import time
t1=time.time()
# notre cacul
t2=time.time()
print(« durée de notre calcul : », t2-t1)
```

Exercice 2 : tri à bulles

Écrire une fonction de tri d'une liste par la méthode du tri à bulles.

La fonction modifie la liste fournie de telle sorte qu'il devienne trié et le retourne.

Le principe est d'inverser les couples de valeurs successives de la liste en fonction de l'ordre du tri, en parcourant la liste d'un bout à l'autre et en répétant l'opération autant de fois que nécessaire.

Exemple avec la liste suivante : 6 10 5 8 12 4

On teste 6 et 10 et on ne fait rien

On teste 10 et 5 et on inverse : 6 5 10 8 12 4

On teste 10 et 8 et on inverse : 6 5 8 10 12 4

On teste 10 et 12 et on ne fait rien

On teste 12 et 4 et on inverse : 6 5 8 10 4 12

Le plus grand (12) est tout en bas.

Et on recommence :

On teste 6 et 5 et on inverse : 5 6 8 10 4 12

On teste 6 et 8 et on ne fait rien

On teste 8 et 10 et on ne fait rien

On teste 10 et 4 et on inverse : 5 6 8 4 10 12

On teste 10 et 12 et on ne fait rien

Et on recommence :

Etc.

Tester votre fonction sur une liste remplie aléatoirement.

3. Matrices

En python, les matrice se déclarent ainsi :

mat=[[1,2,3,4], [5,6,7,8], [9,10,11,12]]

Pour parcourir tous les éléments, on peut faire :

```
for i in range(len(mat)):
    for j in range(len(mat[i])):
        print(mat[i][j])
```

1

Écrire une fonction d'affichage des éléments d'une matrice. Faites-en sorte que l'affichage soit lisible c'est-à-dire, pour l'exemple proposé :

```
 1   2   3   4
 5   6   7   8
 9  10  11  12
```

2

Ecrire une fonction qui calcule la moyenne par ligne.

3

Ecrire une fonction qui calcule la moyenne par colonne.

4

Ecrire une fonction qui multiplie deux matrices.

Avec A matrice n lignes, m colonnes. B matrice m lignes, p colonnes. C matrice n lignes, p colonnes.

Ci,j=somme(k=0, m) Ai,k * Bk,j.

5

Ecrire une procédure qui permet de trier une matrice selon une colonne donnée (utiliser la méthode du tri à bulles).

6

Ecrire une procédure qui permet de trier une matrice selon une ligne donnée (utiliser la méthode du tri à bulles).

7 : matrice – création – tri par colonne ou par ligne

On veut écrire un programme qui permette :

De créer une matrice et de l'afficher

De trier la matrice selon une colonne donnée et aussi selon une ligne donnée.

De recommencer autant de fois qu'on veut.

On pourra choisir le nombre de lignes et le nombre de colonnes.

On pourra choisir de remplir la matrice de 1 à N ou de la remplir avec des nombres choisis aléatoirement dans une plage de valeurs qu'on pourra donner.

8 : carré magique

Un carré magique est un carré divisé en cellules dans lesquelles les nombres entiers, à partir de 1, sont disposés de telle sorte que les sommes de chaque ligne, de chaque colonne et de chaque diagonale soient égales.

Exemple :

4	9	2	
3	5	7	somme = 15
8	1	6	

Plusieurs algorithmes permettent d'obtenir des carrés magiques d'ordre impair (la somme est impaire). Voici le plus simple d'entre eux :

L'élément juste en dessous du centre est occupé par le nombre 1 ;

Les éléments suivants (2, 3, …) sont placés dans les cases se trouvant à l'intersection de la ligne du dessous et de la colonne de droite.

Arrivé au bord du carré, on poursuit l'opération à l'extrémité opposé en suivant la même règle.

Si une case est déjà remplie, le nombre suivant est placé dans la même colonne, deux lignes en dessous.

Ecrire un programme qui lit la taille du coté du carré, engendre le carré magique et l'affiche proprement à l'écran.

4. Problème 1 : Tours de Hanoi

➢ *Description du jeu*

Le jeu est constitué d'une plaquette de bois où sont plantées trois tiges : les tours.

Sur ces tiges sont enfilées des disques de diamètres tous différents.

Les seules règles du jeu sont que l'on ne peut déplacer qu'un seul disque à la fois, et qu'il est interdit de poser un disque sur un disque plus petit.

Au début tous les disques sont sur la tige de gauche, et à la fin sur celle de droite.

On trouve sur internet de nombreuses descriptions du jeu.

➢ *Méthode de résolution itérative*

Si le nombre de disques est impairs :

Déplacer le 1 à gauche (« faire le tour » et revenir sur la tour de droite si on est déjà complètement à gauche).

Sinon // le nombre de disque est pair

Déplacer le 1 à droite (« faire le tour » et revenir sur la tour de gauche si on est déjà complètement à doite).

Déplacer celui des deux qui n'est pas 1 et qu'on peut déplacer.

Répéter les deux opérations jusqu'au déplacement complet de la tour.

➢ *Premier objectif*

Structurer les données de l'exercice.

Écrire une fonction qui affiche les 3 tours. Faire un affichage CLI très simple.

➢ *Deuxième objectif*

Écrire une fonction qui déplace le 1 à gauche

Écrire une fonction qui déplace le 1 à droite

Écrire une fonction qui déplace celui des deux qui n'est pas 1 et qu'on peut déplacer.

➢ *Troisième objectif*

Écrire une fonction qui part de la situation de départ (tous les disques à gauche), et l'affiche, puis qui affiche toutes les étapes pour arriver à la situation finale (tous les disques à droite).

A chaque coup, on affichera le numéro du coup pour arriver au nombre de coups joués pour arriver au but.

Ecrire le main qui utilise cette fonction.

➢ *Méthode*

1) Bien comprendre le fonctionnement du jeu et le principe de résolution : faites-le « tourner » à la main. On peut tester le jeu en ligne, par exemple ici.
2) Quelles données envisagez-vous d'utiliser pour traiter le problème ? (1er objectif)
3) Quel algorithme général pouvez-vous écrire qui traduise le principe de résolution et l'objectif, et qui utilise vos données ? L'algorithme général doit utiliser des fonctions (3ème objectif)
4) Une fois l'algorithme général conçu, il faut écrire le détail de chaque fonction, éventuellement en créant d'autres fonctions (2ème objectif).

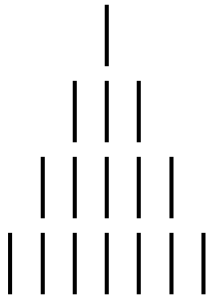
➔ Vous voyez qu'on peut faire les objectifs 2 et 3 dans un sens ou dans l'autre. Au choix.

5. Problème 2 : Jeu des allumettes ou jeu de Marienbad

Présentation

- ***Description du jeu : test ici***

  On a 4 lignes d'allumettes avec 1, 3, 5 et 7 allumettes.



  Le but du jeu est de ne pas ramasser la dernière allumette : qui prend la dernière perd.

  Le jeu se joue à 2. A son tour, chaque joueur peut prendre autant d'allumettes qu'il veut mais sur une seule ligne.

  On peut faire un jeu à 2 ou un jeu contre l'ordinateur.

- ***Comment gagner à tous les coups ?***

  Quand on joue, il faut laisser le jeu dans une configuration particulière expliquée maintenant.

  Chaque ligne contient un nombre d'allumettes : 1, 3, 5 et 7 au départ. Il faut traduire ce nombre en binaire puis sommer les colonnes en base 10.

  Situation de départ :

Nombre d'allumettes	Nombre d'allumettes en binaire		
1	0	0	1
3	0	1	1
5	1	0	1
7	1	1	1
Total en base 10 de chaque colonne binaire :	2	2	4

  Le total en base 10 sera appelé le « T224 ». Le T224 vaut 223 si on retire une allumette !

  Le T224 est un nombre tel que chacun de ses chiffres est pair. On dit alors : un T224 tout pair.

  Pour gagner, il laisser une configuration de jeu avec un T224 tout pair, ce qui n'est possible qu'en partant d'un T224 impair.

  Donc, puisqu'au départ, le T224 est pair, le joueur qui commence va perdre si son adversaire ne fait pas d'erreur, car le joueur qui commence, quoi qu'il fasse, laissera un T224 impair.

  Il restera à son adversaire à jouer pour refaire un T224 tout pair.

- ***Vérifier que vous avez compris comment gagner***

  Essayer de jouer en appliquant la méthode pour gagner à tous les coups pour vérifier que vous avez compris la méthode.

Présentation

- ***Coder un programme qui permette de jouer un joueur contre un autre***

  Utilisez un affichage textuel au choix (un I pour une allumette, par exemple).

  Vous pouvez améliorer l'affichage, tout en restant en mode textuel, à condition de bien encapsuler ça dans des fonctions.

  C'est assez facile : il faut créer des fonctions :

  - pour saisir les informations du coup à jouer,
  - pour mettre à jour le plateau de jeu,
  - pour afficher le plateau de jeu.

- ***Coder un programme qui permette de jouer contre l'ordinateur***

  Faire jouer l'ordinateur est le plus compliqué.

  Une solution sera de tester tous les coups possibles et de jouer dès qu'un coup est gagnant.

  Pour savoir si un coup est gagnant, une technique de calcul pourrait aussi consister à avoir une fonction qui calcule le T224 à partir du jeu (le nombres des allumettes sur chaque ligne). Et avoir une fonction qui nous dit si le T224 est tout pair (ou si le jeu est gagnant).

  Il y a 2 cas particuliers pour l'ordinateur :

  S'il peut gagner, il doit jouer le coup gagnant en priorité.

  S'il peut produire une situation à 3 lignes de 1 (qui est impair, mais gagnante), il doit jouer ce coup en 2ème priorité.

  Ensuite, il applique la règle du T224 tout pair.